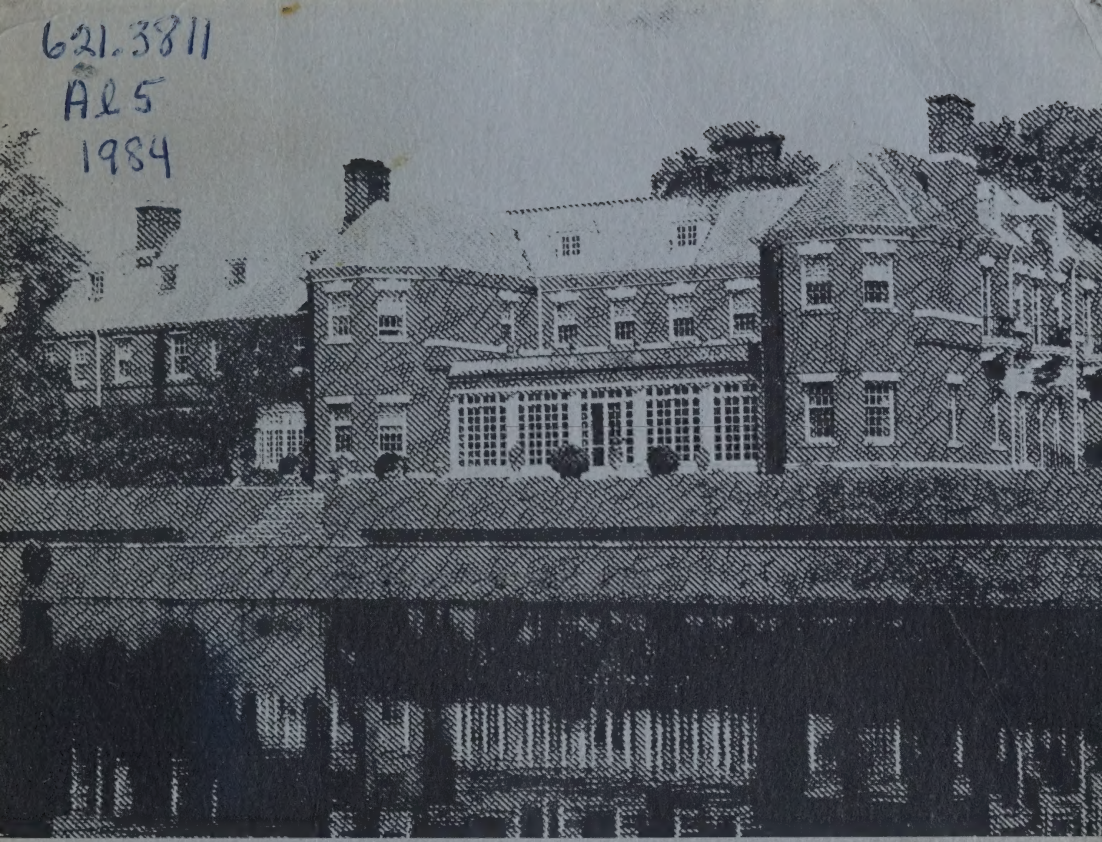
621.3811
A25
1984



Proceedings

TWENTY-SECOND ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

October 3-5, 1984

Allerton House, Monticello, Illinois
Sponsored by the
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering of the
University of Illinois at Urbana-Champaign

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

To renew call Telephone Center, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

ENGINEERING

INTERLIBRARY LOAN DEC 5 RECD

MAY 23 RECD

INTERLIBRARY LOAN

MAY 6 RECD

JAN 13 1993

MAR 04 RECD

INTERLIBRARY LOAN

JUL 02 RECD

NOV 25 1987

NOV 02 RECD

INTERLIBRARY LOAN

DATE 12-5-86

L161—O-1096

PROCEEDINGS

TWENTY-SECOND ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

David C. Munson, Jr.
Tamer Başar
Conference Co-Chairmen

Conference held
October 3-5, 1984
Allerton House
Monticello, Illinois

Sponsored by the
Coordinated Science Laboratory
and the
Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

621.3811
A45
1984
Engen

FOREWORD

The Twenty-Second Annual Allerton Conference on Communication, Control, and Computing was held at Allerton House, Monticello, Illinois on October 3, 4, and 5, 1984. The Conference was sponsored by the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign. These PROCEEDINGS contain the papers presented at the Conference.

In addition to the large number of regular sessions, there was an invited Plenary Session, and also an invited special session on Radio Networks. We are grateful to Professor Thomas Kailath of Stanford University for delivering the plenary lecture entitled, "Digital Filtering in VLSI." Our thanks are also extended to Professor Fouad Tobagi of Stanford University for organizing and chairing the special session on Radio Networks. Their time and efforts on behalf of the Conference are deeply appreciated. We would also like to thank the other session chairmen and authors for their contributions to the success of the Conference.

The support of both Professor T. N. Trick, Acting Director of the Coordinated Science Laboratory, and Professor E. W. Ernst, Acting Head of the Department of Electrical and Computer Engineering is appreciated and acknowledged. We would also like to thank our colleagues, Professors J. A. Abraham, T. Ü. Başar, J. B. Cruz, Jr., B. E. Hajek, I. N. Hajj, T. S. Huang, W. K. Jenkins, P. V. Kokotovic, H. Lee, M. C. Loui, J. Medanic, J. H. Patel, W. R. Perkins, H. V. Poor, F. P. Preparata, M. B. Pursley, V. Ramachandran, and D. V. Sarwate for their generous assistance in organizing the Conference.

We are grateful to Mr. Steven Williams for his handling of the business affairs of the Conference and to Ms. Rose Harris for handling of registration both prior to and during the Conference. Finally, very special thanks are due to Ms. Rosemary Wegeng for her capable and good natured assistance in all aspects of the organization of the Conference, from the initial preparation of the Conference announcement, through Conference registration, and to the final editing of these Proceedings.

David C. Munson, Jr.

Tamer Başar

TABLE OF CONTENTS

FOREWORD iii

PLENARY SESSION

Chairman: M. T. Başar
University of Illinois

DIGITAL FILTERING IN VLSI 1
S. K. Rao and T. Kailath

I-A: RADIO NETWORKS

Organizer and Chairman: Fouad A. Tobagi
Stanford University

PERFORMANCE EVALUATION OF A MULTIPLE ACCESS SYSTEM USING AN APPROXIMATE MODEL 11
R. Zhu and A. Ephremides

THROUGHPUT ANALYSIS OF MULTIPLE HOP PACKET RADIO NETWORKS USING A CODE DIVISION MULTIPLE ACCESS (CDMA) PROTOCOL 20
M.-S. Chen, R. Boorstyn, and A. Kershenbaum

PERFORMANCE OF SPREAD SPECTRUM NETWORKS 30
J. A. Silvester

IMPROVEMENTS IN THROUGHPUT OF A CDMA PACKET RADIO NETWORK DUE TO A CHANNEL LOAD SENSE ACCESS PROTOCOL . . . 40
F. A. Tobagi and J. S. Storey

A CHANNEL ACCESS PROTOCOL FOR SURVIVABLE RADIO NETWORKS WITH FREQUENCY HOPPING SPREAD SPECTRUM SIGNALING 50
D. Baker, J. Wieselthier, and A. Ephremides

ON THE THEORY OF DISTRIBUTED PROCESSING 60
L. Kleinrock

I-B: COMPUTATIONAL GEOMETRY

Chairman: Franco P. Preparata
University of Illinois

CRITICALITY CONSIDERATIONS IN THE DESIGN OF GEOMETRIC ALGORITHMS 71
B. Chazelle

FINDING RESTRICTED K-ENVELOPES FOR CONVEX POLYGONS 81
N. A. A. DePano and A. Aggarwal

INSPECTION BY POLYGON CONTAINMENT 91
B. S. Baker, S. J. Fortune and S. R. Mahaney

DETECTING SYMMETRY IN A PLANAR FIGURE 101
M. J. Atallah

SHORTEST PATHS BETWEEN SOURCE AND GOAL POINTS LOCATED ON/AROUND A CONVEX POLYHEDRON 103
W. R. Franklin and V. Akman

DRAWING NONINTERSECTING SEGMENTS BETWEEN TWO PLANAR SETS 113
M. J. Atallah

AN ALGORITHM FOR COMPUTING ORDER k VORONOI DIAGRAMS IN THE PLANE 120
B. Bhattacharya

I-C: ADAPTIVE AND NONLINEAR SIGNAL PROCESSING

Chairman: W. Kenneth Jenkins
University of Illinois

AN ADAPTIVE MINIMUM-VARIANCE PREDICTOR-SMOOTHER ALGORITHM FOR IMAGE ENHANCEMENT 122
C. Rey and R. Ward

A SUBOPTIMAL SOLUTION TO IIR ADAPTIVE FILTERING USING AN EXTENDED LMS ALGORITHM 132
E. Fernandez-Gaucherand and J. Y. Cheung

ADAPTIVE NOISE CANCELLING AND NONLINEAR ESTIMATION THEORY 142
A. Abutaleb

ADAPTIVE NONLINEAR DIGITAL FILTER WITH SEQUENTIAL REGRESSION ALGORITHM 152
D. H. Youn, K. K. Yu, and V. J. Mathews

THEORETICAL ANALYSIS OF THE MAX/MEDIAN FILTER 162
G. R. Arce and M. P. McLoughlin

A MEDIAN FILTER ARCHITECTURE SUITABLE FOR VLSI IMPLEMENTATION 172
G. R. Arce and P. J. Warter

STATISTICAL PROPERTIES OF THE ANALOG MEDIAN FILTER . . . 182
J. P. Fitch, E. J. Coyle, and N. C. Gallagher

I-D: DECENTRALIZED CONTROL AND MODEL REDUCTION TECHNIQUES

Chairman: Marija Ilic-Spong
University of Illinois

DECENTRALIZED CONTROL FOR A CLASS OF NONLINEAR SYSTEMS USING A VARIABLE STRUCTURE APPROACH 192
G. P. Matthews, S. H. Zak, and R. A. DeCarlo

DECENTRALIZED CONTROL USING QUASI BLOCK DIAGONAL DOMINANCE OF TRANSFER FUNCTION MATRICES 201
Y. Ohta, D. D. Siljak, and T. Matsumoto

SIMULTANEOUS ROUTING AND SIGNALIZATION IN TRAFFIC NETWORKS 211
F. Khorrami and U. Ozguner

ANALYSIS OF REDUCED ORDER TWO-TIME-SCALE MODELS 221
S. J. Fu, M. E. Sawan, and M. T. Tran

THE FIXED POINT NUMERICAL SOLUTION OF ALGEBRAIC RICCATI EQUATION OF SINGULARLY PERTURBED SYSTEMS 223
Z. Gajic

A COMPOSITE CONTROL STRATEGY FOR A WEAKLY ELASTIC ROBOT 225
A. Ficola, R. Marino, and S. Nicosia

REDUCED ORDER MODELING AND FINITE DIMENSIONAL STABILIZATION OF LINEAR DISTRIBUTED PARAMETER SYSTEMS VIA GALERKIN'S TECHNIQUE 232
J. G. Hsieh and M. J. Balas

II-A: DATA STRUCTURES AND GRAPH ALGORITHMS

Chairman: Vijaya L. Ramachandran
University of Illinois

A LOCALLY ADAPTIVE DATA COMPRESSION SCHEME 233
J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei

ON THE USE OF SMOOTHSORT AND STOCHASTIC MOVE-TO-FRONT OPERATIONS FOR OPTIMAL LIST ORGANIZATION 243
B. J. Oommen

MANAGING STORAGE FOR EXTENDIBLE ARRAYS OF VARYING DIMENSIONS 253
E. J. Otoo

DICTIONARIES ON BOUNDED DOMAINS 264
R. G. Karlsson

ON-LINE UPDATING OF DEGREE-CONSTRAINED MINIMUM SPANNING TREES 271
G. N. Frederickson and M. A. Srinivas

ON THE PROBABLE PERFORMANCE OF GRAPH COLORING ALGORITHMS 281
J. S. Turner

FEEDBACK VERTEX SETS IN POLYNOMIAL TIME - A NEW CLASS 291
E. L. Lloyd, M. L. Soffa, and C.-C. Wang

II-B: SIGNAL DETECTION AND ESTIMATION

Chairman: H. Vincent Poor
University of Illinois

ROBUST ESTIMATION OF AR MODELS USING THE LATTICE STRUCTURE 299
S. Li and B. W. Dickinson

RANDOMIZED ROBUST CONFIDENCE PROCEDURES 309
K. Martin and M. Mintz

STABLE LINEAR DETECTORS WITH OPTIMALITY CONSTRAINTS . . . 318
P. A. Kelly and W. L. Root

ON THE OPTIMALITY OF DATA PROCESSORS FOR SIGNAL DETECTION OVER A CLASS OF CONTAMINATED NOISES 328
D. R. Halverson and G. L. Wise

GEOMETRICAL APPROACH TO GENERALIZED LEAST SQUARES ESTIMATION WITH SYSTOLIC ARRAY PROCESSING 333
S. Kalson and K. Yao

ESTIMATING THE INTENSITY FUNCTION OF A POISSON PROCESS WHEN EACH OF ITS POINTS IS OBSERVED AFTER A RANDOM TRANSLATION 343
D. L. Snyder

LARGE ERROR PERFORMANCE OF CROSS-CORRELATION TYPE ESTIMATORS OF TIME DELAY 344
A. Hero and S. Schwartz

DETECTION OF SIGNALS WITH RANDOM ARRIVAL TIME 354
L. A. Longley and C. C. Lee

II-C: ADAPTIVITY AND STABILITY IN CONTROL SYSTEMS

Chairman: P. R. Kumar
University of Maryland, Baltimore Co.

ON ADAPTIVE PARAMETER IDENTIFICATION OF LINEAR SYSTEMS WITH APPLICATION TO ADAPTIVE CONTROL/ OBSERVERS 364
E. Emre

ON MODEL REFERENCE ADAPTIVE CONTROL (MRAC) AND IDENTIFICATION - ALGORITHMS TO EASE IMPLEMENTATION 374
P. J. Kennedy, Jr.

CONTINUOUS STABILIZATION OF NONLINEAR AND FAMILIES OF LINEAR TIME-VARYING SYSTEMS 376
E. Emre and G. Knowles

II-D: COMMUNICATION NETWORKS

Chairman: Anthony Ephremides
University of Maryland

DELAY ANALYSIS OF THE n-ARY STACK ALGORITHM FOR A RANDOM-ACCESS BROADCAST CHANNEL 385
L. Merakos and C. Bisdikian

THE DISTRIBUTION OF DELAY AND CAUDAL CHARACTERISTIC CURVE OF CSMA/CD NETWORKS 395
S. L. Beuerman and E. J. Coyle

TWO-COMMODITY NETWORK DESIGN WITH MULTIPLE DEMAND 405
M. Lucertini and G. Paletta

A PERFORMANCE COMPARISON OF THE TOKEN RING, TOKEN BUS, AND CSMA/CD BUS 413
C. T. A. Lea and J. S. Meditch

III-A: TOPICS IN DIGITAL SIGNAL PROCESSING

Chairman: David C. Munson, Jr.
University of Illinois

A FAST ALGORITHM TO OPTIMIZE THE MULTIPLIER COEFFICIENTS OF CERTAIN CLASSES OF DIGITAL FILTERS 421
M. Toy and P. M. Chirlian

STABILITY ANALYSIS OF FIXED-POINT DIGITAL FILTERS USING COMPUTER GENERATED LYAPUNOV FUNCTIONS: WAVE AND LATTICE FILTERS 431
K. T. Erickson and A. N. Michel

ISOMORPHISM BETWEEN THE CONTINUOUS SPACES SIGNALS AND DISCRETE SPACES SIGNALS.COMPLEMENTS 441
C. Bozzo and G. Bueno

A ROBUST CONJUGATE DIRECTIONS METHOD FOR SOLVING LINEAR SYSTEMS 450
B. J. Sullivan and B. Liu

OPTIMUM AMPLITUDE WEIGHTING FOR RANGE SIDELOBE SUPPRESSION IN A HIGH RESOLUTION RADAR HYBRID WAVEFORM DESIGN 459
D. B. Koch and W. H. Tranter

ARRAY FILTERS FOR THE ATTENUATION OF A COHERENT UNDESIRED SIGNAL IN THE PRESENCE OF WHITE NOISE 469
M. T. Hanna and M. Simaan

III-B: COMBINATORIAL ALGORITHMS AND VLSI

Chairman: Donna J. Brown
University of Colorado and
University of Illinois

BREAKING A PARAGRAPH INTO LINES IN LINEAR TIME 478
L. L. Larmore and D. S. Hirschberg

A FAST LINE INTERSECTION ROUTING METHOD FOR OPTIMAL WIRINGS 488
J. P. Cohoon

A 1.6 APPROXIMATION ALGORITHM FOR ROUTING MULTITERMINAL NETS AROUND A RECTANGLE 493
T. F. Gonzalez and S.-L. Lee

AVERAGE CASE ANALYSIS OF MARKING ALGORITHMS 508
D. S. Hirschberg and L. L. Larmore

THE CLOSEST NEIGHBOR PROBLEM 510
W. Cunto and J. I. Munro

A FULLY-POLYNOMIAL APPROXIMATION SCHEME FOR VARIABLE SIZED BIN PACKING 516
F. D. Murgolo

THE DESIGN AND ANALYSIS OF BUCKETSORT 525
E. E. Lindstrom and J. S. Vitter

III-C: CODING AND QUANTIZATION

Chairman: Richard E. Blahut
IBM Corporation and
University of Illinois

UNDETECTED ERROR PROBABILITY AND THROUGHPUT ANALYSIS OF A CONCATENATED CODING SCHEME 535
H. Deng and D. J. Costello, Jr.

USING SIMULATED ANNEALING TO CALCULATE COMBINATORIAL CONSTANTS 545
L. Hemachandra and V. K. Wei

NUMBER OF INFORMATION SYMBOLS OF BCH CODES OF LENGTH 2^m+1 553
A. Caceres and O. Moreno

TRANSFORM APPROACH TO GOPPA CODES 559
M. Loeloeian and J. Conan

TREE CODING OF SPEECH WITH BLOCK-ADAPTIVE FILTERS . . . 560
W. A. Pearlman and M. Johnson

HYBRID TRANSFORM/DPCM CODING OF IMAGES USING PYRAMID VECTOR QUANTIZATION 569
T. R. Fisher and H.-C. Tseng

SOME COMMENTS ON THE DESIGN OF QUANTIZERS 579
E. F. Abaya and G. L. Wise

III-D: STOCHASTIC SYSTEMS

Chairman: Tangül Başar
University of Illinois

ON SELF-TUNING TO THE OPTIMAL CONTROLLER 587
P. R. Kumar

ROBUST CONTROL OF LINEAR STOCHASTIC SYSTEMS WITH UNCERTAIN DYNAMICS: A GAME-THEORETIC APPROACH 588
E. Ryan-Schuck and M. Mintz

A NON-LINEAR LEAST SQUARE ALGORITHM FOR PARAMETER IDENTIFICATION 597
J. Gertler and C. Ferreira

MOVING HORIZON STABILIZATION OF RANDOM PARAMETER SYSTEMS 599
E. Yaz

TIME OPTIMAL SEMI-MARKOV DECISION PROCESSES UNDER A CONSTRAINT 603
F. J. Beutler and K. W. Ross

ON THE INSTRUMENTAL VARIABLE METHOD 613
S. Puthenpura and N. K. Sinha

III-E: COMPENSATOR DESIGN FOR DYNAMIC SYSTEMS

Chairman: Juraj Medanic
University of Illinois

FINITE DIMENSIONAL COMPENSATOR DESIGN FOR LINEAR RETARDED SYSTEMS 615
W.-S. Lu, E. B. Lee, and S. H. Zak

SIMULTANEOUS STABILIZATION, SENSITIVITY MINIMIZATION, TRACKING AND DISTURBANCE REJECTION BY INTERPOLATION METHODS 625
B. K. Ghosh

COMPENSATOR PARAMETRIZATION DESIGN FOR MULTIVARIABLE SYSTEMS 635
S.-Y. Zhang

COMPUTING THE ESTIMATOR-CONTROLLER FORM OF A COMPENSATOR 642
D. J. Bender and R. A. Fowell

DESIGN OF MINIMAL MULTIFUNCTIONAL OBSERVER 652
B. Shafai and R. L. Carroll

IV-A: PARALLEL AND DISTRIBUTED COMPUTATION

Chairman: Michael C. Loui
University of Illinois

A PARALLEL ALGORITHM FOR ALL PAIRS SHORTEST PATHS IN A RANDOM GRAPH 663
A. Frieze and L. Rudolph

PARALLEL APPROXIMATION SCHEMES FOR SUBSET SUM AND KNAPSACK PROBLEMS 671
J. Peters and L. Rudolph

O(log N) PARALLEL ALGORITHMS FOR BINARY TREE TRAVERSALS 681
K. V. S. Ramarao

SIMPLE AND EFFICIENT DISTRIBUTED ALGORITHMS FOR ELECTION IN COMPLETE NETWORKS 689
Y. Afek and E. Gafni

DISTRIBUTED ELECTION IN UNIDIRECTIONAL EULERIAN NETWORKS 699
E. Gafni and W. Korfhage

PROBLEMS IN BROADCASTING AND GOSSIPING 701
D. Richards and A. L. Liestman

IV-B: COMMUNICATION SYSTEMS

Chairman: Wayne E. Stark
University of Michigan

FH-MFSK SYSTEMS 711
G. E. Atkin and I. F. Blake

ERROR PERFORMANCE ANALYSIS OF RANKING RECIVER AND A NON-LINEAR SQUARE LAW RECEIVER FOR FH-BFSK TRANSMISSION IN WORST-CASE PARTIAL BAND JAMMING 720
R. Viswanathan, K. Taghizadeh, and R. Nerella

NEW MODULATION CODES FOR MAGNETIC RECORDING 730
C. Heegard and T.-A. Lee

A HETERODYNE RECEIVER FOR AN OPTICAL COMMUNICATION SYSTEM WITH AN UNSTABLE LASER LOCAL OSCILLATOR 739
C. N. Georghiades and D. L. Snyder

A COMPARISON OF BOUNDS AND SIMULATION RESULTS ON ERROR PROPAGATION IN DECISION FEEDBACK RECEIVERS 741
J. J. O'Reilly, A. M. De Oliveira Duarte, and T. Beck

CODING PERFORMANCE FOR A DIRECT-DETECTION OPTICAL FIBER CHANNEL 743
M. A. Herro and L. Hu

IV-C: LINEAR SYSTEMS: ANALYSIS AND SPECTRUM ASSIGNMENT

Chairman: S. P. Bingulac
VPI and University of Belgrade, Yugoslavia

THE DUAL GHR 745
D. K. Lindner

INVARIANT MANIFOLDS FOR DISCRETIZED SYSTEMS 753
D. L. Elliott

ZERO ASSIGNMENT WITH MEASUREMENT FEEDBACK 755
S. O. Majdi, J. W. Howze and S. P. Bhattacharyya

OPTIMIZATION OF DECENTRALIZED EIGENVALUE ASSIGNMENT . . . 757
D. Sebok and R. A. DeCarlo

POLE ASSIGNMENT BY USING DYNAMIC OUTPUT FEEDBACKS 759
S.-Y. Zhang

ON A POLE ASSIGNMENT PROPERTY OF KLEINMAN'S STABILIZATION RULE 761
H. Selbuz and E. Yaz

STATE FEEDBACK DESIGN FOR ROBUST MULTIVARIABLE SYSTEMS 763
N. K. Sinha and P. Rozsa

IV-D: IMAGE PROCESSING

Chairman: Jan P. Allebach
Purdue University

PARALLEL COMPUTATION OF NORMALIZED FOURIER DESCRIPTORS 765
K. D. Smith and L. H. Jamieson

MARKOV MODEL FOR IMAGE SEGMENTATION 775
N.-K. Huang

SEQUENTIAL EDGE LINKING 782
P. H. Eichel and E. J. Delp

SPLINE BASED SEQUENTIAL ESTIMATION OF BOUNDARIES IN DIGITAL IMAGES 792
A. Liakopoulos and D. Brzakovic

RECONSTRUCTION OF REAL 2-D SIGNALS FROM THEIR IN-LINE HOLOGRAMS BY APPROXIMATE INVERSE FILTERING 794
L. Onural and P. D. Scott

A HOLOGRAM-LIKE DIGITAL REPRESENTATION OF PICTURES . . . 796
P. Prusinkiewicz

RECONSTRUCTION OF 2-D SIGNALS FROM IRREGULARLY SPACED SAMPLES WITH AN APPLICATION TO THE RECOVERY OF TIME-SEQUENTIALLY SAMPLED MOVING OBJECTS 798
D. S. Chen and J. P. Allebach

AN IMPROVED APPROXIMATION IMAGE FOR MULTIPLE SOURCE CODES 808
T. Papoutsis, L. F. Chaparro, and M. Kanefsky

IV-E: TESTING AND FAULT TOLERANCE

Chairman: Jacob A. Abraham
University of Illinois

FUNCTIONAL TESTING OF A MICROPROGRAMMED PROCESSOR DESIGNED FOR TESTABILITY 817
F. Ozguner, J. K. Wang, K. W. Olson, and E. Ribble

TESTABLE DESIGN OF SEQUENTIAL MACHINES USING COMPACTION FUNCTIONS AND CHECKING EXPERIMENT 827
K. K. Saluja and R. Dandapani

ANALYSIS OF ALTERNATIVE IMPLEMENTATIONS OF PARALLEL SIGNATURE ANALYZERS 837
P. C. Maxwell

TEST DERIVATION FOR STUCK-OPEN FAULTS IN ITERATIVE LOGIC ARRAYS 844
R. Sharma and D. Radhakrishnan

A FAULT TOLERANT PROCESSOR SWITCH ARRAY STRUCTURE FOR WAFER SCALE INTEGRATION 854
A. D. Singh

V-A: NONLINEAR CONTROL SYSTEMS AND APPLICATIONS

Chairman: Mark W. Spong
University of Illinois

NONLINEAR FEEDBACK DESIGN: OPTIMAL RESPONSES BY TENSOR ANALYSIS 864
J. A. O'Sullivan and M. K. Sain

COMPUTED REFERENCE ERROR ADJUSTMENT TECHNIQUE (CREATE) FOR THE CONTROL OF ROBOT MANIPULATORS 874
S.-H. Wang

DESIGN OF A CONTROL SYSTEM FOR BATCH PRODUCTION ON A DISTILLATION COLUMN 876
M. K. Yurtseven and W. F. Fincham

V-B: MATRIX COMPUTATIONS IN LINEAR SYSTEMS

Chairman: Ibrahim N. Hajj
University of Illinois

FAST AND STABLE ITERATIVE METHODS FOR THE SOLUTION OF POSITIVE DEFINITE TOEPLITZ SYSTEMS OF LINEAR EQUATIONS 878
R. R. Bitmead and G. H. Allen

SYSTEM ORDER ESTIMATION: A SINGULAR VALUE DECOMPOSITION APPROACH 887
L. H. Sibul and J. A. Tague

ON THE ROBUSTNESS OF AN ORDER-ESTIMATION TECHNIQUE . . . 889
J. M. Flentz and Y. F. Huang

A PERTURBATION APPROACH TO IMPROVING PISARENKO HARMONIC RETRIEVAL 891
D. R. Fuhrmann and B. Liu

IMPLEMENTATION AND ANALYSIS OF A COMPLEX-VALUED SVD ALGORITHM 901
K. Yao, T. Y. Yan, and E. Biglieri

V-C: FOUNDATIONS OF COMPUTING

Chairman: David A. Plaisted
University of Illinois

LOWER BOUNDS ON THE MONOTONE NETWORK COMPLEXITY OF THRESHOLD FUNCTIONS 911
P. E. Dunne

EDGE GRAMMARS: DECIDABILITY RESULTS AND FORMAL LANGUAGE ISSUES 921
F. Berman and G. Shannon

DECIDING LIVENESS FOR SPECIAL CLASSES OF COMMUNICATING FINITE STATE MACHINES 931
C.-K. Chang, M. G. Gouda, and L. E. Rosier

V-D: SPECIAL PURPOSE PROCESSORS

Chairman: Janak H. Patel
University of Illinois

A SYSTOLIC ARCHITECTURE BASED ON ADDERS FOR DIGITAL SIGNAL PROCESSOR 940
M. A. Haddad and P. A. Ramamoorthy

SYSTOLIC SORTING IN A SEQUENTIAL INPUT/OUTPUT ENVIRONMENT 946
S. G. Akl and H. Schmeck

A REAL-TIME 2-LEVEL SYSTOLIC 2-D CONVOLUTION CHIP 956
C. Bozzo, M. Fouques, and E. Siffredi

SYSTOLIC ALGORITHMS FOR LOCAL OPERATIONS ON IMAGES . . . 965
C. Guerra

V-E: INFORMATION THEORY

Chairman: Chris Heegard
Cornell University

CODES GENERATED BY PERMUTATIONS FOR DISCRETE MEMORYLESS MULTIPLE-ACCESS CHANNELS WITH SLEPIAN-WOLF CORRELATED SOURCES 975
K. DeBruyn

TWO-DIMENSIONAL BLOCK INTERFERENCE CHANNELS WITH APPLICATIONS TO RANDOM ACCESS MEMORIES 983
W. E. Stark and J. H. Lee

THE CAPACITY OF JAMMED COMMUNICATIONS CHANNELS USING SPREAD SPECTRUM MODULATION 992
B. L. Hughes and P. Narayan

A RESULT ON THE CHARACTERIZATION OF THE THREE-DIMENSIONAL CAPACITY REGIONS, WITH AND WITHOUT FEEDBACK, OF THE BINARY ERASURE MULTIPLE-ACCESS CHANNEL 997
K. DeBruyn and E. C. van der Meulen

DIGITAL FILTERING IN VLSI†

Sailesh K. Rao and Thomas Kailath

Information Systems Laboratory
Stanford University
Stanford, CA 94305.

ABSTRACT

There are certain desirable criteria for architectural implementations using VLSI technology. The tailoring of algorithms and signal flow graphs to meet these criteria has recently received considerable attention. For a certain important class of digital filter circuits, it is shown in this paper that the properties of the implementation that are relevant to VLSI can be predicted by using some parameters of the signal flow graph of the circuit, without taking recourse to CAD tools; classical concepts and design algorithms from network theory can then be suitably extended and generalized to provide a hierarchical synthesis procedure that imposes the various desirable criteria in a step-by-step fashion. At each step, the sub-class of digital filter circuits that meet the imposed criteria can be completely parametrized.

INTRODUCTION

In recent years, technological advances have led to major developments in the areas of custom VLSI chip design, digital signal processors and general-purpose parallel computers. For a given task, there can be a myriad of algorithms and implementations that accomplish it, and for a given algorithm, there are, almost invariably, certain architectural or processor configurations that execute the algorithm efficiently. Thus, as an algorithm designer, it is of importance to know what sort of algorithms would lead to efficient custom VLSI chip designs and/or which class of algorithms is tailor-made for utilizing to the fullest extent possible, the capabilities of the signal processor or the parallel computer at one's disposal. On the other hand, designers of a parallel computer for example, would be interested to know the nature of the hard-wired inter-processor communications that they need to allow and the software tools that they need to provide in order to efficiently accomplish the various tasks that the machine is intended for.

In the above setting, the problem of digital filtering has been addressed by several authors. For example, researchers at KU Leuven, Bochum and Siemens [1-4], have devised algorithms for producing custom VLSI chips for a digital filter that is assumed to be specified in the form of a signal flow graph. An interesting aspect of their solution is that two circuits with identical topological configurations but with different parameters can lead to different layouts at the mask level due to the various optimization steps executed in the process. In another direction, researchers at UC Berkeley [5] have devised CAD tools for mapping a given signal flow graph into an optimal software code for a special purpose programmable signal processor generator. In both cases, one common objective, for example, that is considered by the authors is the minimization of the sampling interval of the resulting circuit; they have found, and it can be shown from elementary circuit theoretic arguments, that this sampling interval is limited by the inherent sequentiality of the signal flow graph that the circuit is intended to simulate.

Several studies have been reported in the literature regarding this limitation. In these studies, the authors have attempted to 'systolicize' a given signal flow graph via cut-set delay transformations [6,7], or via schedule-assignment methods [8], in order to predict the minimum sampling interval that can be achieved by the circuit as a multiple of the internal clock period. Using their techniques, it can be shown that two different signal flow graph realizations of the same transfer function can lead to different sampling intervals. In certain extreme cases, it can be shown that there is negligible improvement in the throughput of the circuit simulating the signal flow graph using multiple

†This work was supported in part by the US Army Research Office, under Contract DAAG29-83-K-0028, the Air Force Office of Scientific Research, Air Force Systems Command, under Contract AF83-0228, and Lockheed Missile & Space Corp., under Contract FBS6A8930F.

processing units when compared to a sequential implementation on a single processor.

The above discussion together with the fact that there is an infinite number of signal flow graph realizations for a digital filter specified in terms of its transfer function, motivates the following questions:

Are there certain measurable properties of the signal flow graph representation of a circuit that predict its suitability for implementation in VLSI?

If so, is there a systematic way in which realizations that possess these desirable properties can be generated?

In this paper, we attempt to answer the questions raised above. The problem is perceived purely from the viewpoint of an algorithm designer; thus, the aim is to develop synthesis procedures that map the desired input-output characteristics of the filter (assumed to be given in the form of a rational transfer function) into a signal flow graph representation that can be easily identified with an architectural configuration that meets the following criteria:

(1) *Modularity:* The circuit should preferably be a regular interconnection of similar *processing elements.* Processing elements themselves can be identified with sections of the signal flow graph as some interconnection of multipliers, summers, and storage modules that contain the states of the elements. The condition of modularity would then be equivalent to the statement that the sections of the signal flow graph identified with each processing element should have identical topological configurations. The processing elements can, however, be implemented on general-purpose processors where modularity would require that each processor sequences through identical steps (except, maybe, for some changes in the parameters, i.e., register or memory contents) over each iteration. In this paper, it will be assumed that each processing element contains exactly one state.

(2) *Local Communication:* Communication between the processing elements should be restricted to immediate neighbours. A modular array of processing elements that possesses this property will be termed a *VLSI Circuit* or a *VLSI Array.*

(3) *Pipelineability:* In a VLSI circuit, data transfer can occur only between immediate neighbours. Assuming that such a transfer requires at least some finite non-zero time, the circuit will be said to be pipelineable if there exists an achievable computational/data transfer schedule for the operation of the circuit such that the rate at which the input data is processed is independent of the order of the filter n (i.e., of the number of processing elements in the array) for some n greater than a constant.

(4) *Numerical Stability:* There are certain undesirable phenomena, e.g., *limit-cycle and overflow oscillations,* associated with the finite-precision implementation of linear systems that have been well-studied in the literature. The circuit should be free of them. It should also be insensitive to slight variations in the multiplicative parameters in the circuit.

For several reasons (see [9-13]) not outlined here, it will be assumed that the condition of numerical stability of the circuit is synonymous with the condition that the transition matrix of each processing element in the VLSI array is orthogonal.

The approach employed in this paper to solve this problem can be briefly described as follows:

Step 1: Decide upon the interconnecting pattern for the processing elements.

Step 2: Choose a model comprised of adders, multipliers, and storage modules (z^{-1} blocks) for the processing element.

Step 3: Choose suitable terminating conditions for the ends of the array, if necessary.

Step 4: Determine the conditions that have to be imposed upon the parameters of (the model for) each processing element so that the resulting array is a realization for the given transfer function.

Step 5: Among the realizations generated in Step 4, determine the sub-class that leads to pipelineable implementations and parametrize this class.

Step 6: Determine the conditions upon the parameters of the realizations obtained in Step 5 in order to ensure that the transition matrix of each processing element is orthogonal.

Remark: The role of pipelineability and orthogonality can be interchanged in the hierarchical development outlined above. Details can be found in [9].

There are several features that distinguish the work reported in this paper from that found in the literature[14-16]. The results we have obtained using our approach seem to be more general and comprehensive. In [16] the authors have obtained block digital filter implementations that lead to pipelined two-dimensional systolic arrays whose area-complexity is quadratic in both the order of the filter n and the length of the block L. In contrast, we derive recursive implementations ($L = 1$) whose area complexity function grows linearly with the order of the filter.

The hierarchical synthesis procedure described in this paper maps every stable digital transfer function into a signal flow graph with a ***unique topology***. For a given transfer function, there are infinitely many parametrizations of the multiplicative gains associated with the branches of the graph and the resulting circuits all lead to pipelineable architectures wherein the arithmetic operations requested of the processing element are elementary Givens rotations. Based upon the uniqueness of the topology, we have been able to design a signal processor (not yet implemented) whose arithmetic unit is a pipelined CORDIC chain (see [17-19] for details on the CORDIC algorithm and [20] for the pipelined CORDIC chain) and which has two I/O ports in order to enable it to communicate with neighbouring processors. This signal processor can be programmed to perform multiple-input multiple-output linear time-invariant or adaptive-lattice filtering.

MODULAR LOCALLY-INTERCONNECTED REALIZATIONS

For general pole-zero filtering, a multi-rate systolic array was proposed in [21,22]. It consists of n processing elements, where n is the order of the transfer function of the filter, arranged in a linear configuration with only nearest-neighbour interconnections between the elements. The processing elements of the array are identifiable with three-input three-output single-state linear systems and a signal-flow graph type of representation of this systolic array can be shown to be identical to that of the well-known controller canonical form realization described in e.g. [23]. Furthermore, it can be shown [9,22] that the controllability, observer, and observability canonical forms described in [23], the Gray-Markel lattice described in [24] and the orthogonal digital filter structures in [9-13] can be made to operate in a pipelined, systolic fashion in a linear array of processing elements that are three-input three-output single-state linear systems. Using these and many other systolic type arrays derived from existing canonical designs [9], one can arrive at the following common features that characterize these arrays (see Fig.1):

[a] The arrays can be described as a linear interconnection of similarly parametrized processing elements.

[b] Each processing element is a three-input three-output single-state linear system and communicates only with its immediate neighbours; in any particular iteration, this data-transfer between a processing element and its left or right neighbour occurs in both directions (see Fig.2).

[c] The input to the digital filter and the corresponding output are obtained as linear combinations of the signals appearing at one end of the array.

Despite the commonality of these features, it should be noted that the original derivation of the various circuits mentioned above were obtained using different considerations. For example, the controller canonical form realization, (i.e., the multi-rate systolic array found in [21,22]) was obtained by directly interpreting the recurrence relation underlying the filter, while the orthogonal filter structure found in [13] was obtained as a result of solving a certain least-squares optimization problem; the cascade form realization was obtained using a pole-zero decomposition of the transfer function and the parallel form realization was obtained using a partial fraction expansion of the transfer function. Since all these circuits can be described in terms of the linear array of Fig.4, the question arises as to whether or not there exists a general methodology that unifies the derivation of these circuits and as to whether hitherto unknown parametrizations of the array exist that solve the same problem. The answers to both these questions are in the affirmative as substantiated in what follows.

Let $H(z) = N(z)/D(z)$ be the transfer function of the digital filter to be synthesized where $N(z)$, $D(z)$ are polynomials in z with $D(z)$ of degree n and $N(z)$ of degree at most n.

Suppose that the designer has available a linear array of processing elements as in Fig.1 and that the processing elements are capable of simulating the signal flow graph of Fig.2. In terms of this representation, let the circuit of Fig.3 be a realization for the transfer function $H(z)$.

Define the *transfer matrix* of each processing element, $\Sigma(z)$, in Fig.2 to be

$$\Sigma(z)\begin{bmatrix}u_1(z)\\u_2(z)\\u_3(z)\end{bmatrix} = \begin{bmatrix}y_1(z)\\y_2(z)\\y_3(z)\end{bmatrix}; \quad \Sigma(z) = \begin{bmatrix}\frac{CB^T}{z-a} + D\end{bmatrix} = \begin{bmatrix}\sigma_{11}(z) & \Sigma_{12}^T(z)\\ \Sigma_{21}(z) & \Sigma_{22}(z)\end{bmatrix} \tag{1}$$

where $\{a, B^T, C, D\}$ are constant matrices with finite elements and of dimensions (1×1), (1×3), (3×1), (3×3) respectively and $\Sigma(z)$ is appropriately block-partitioned such that $\sigma_{11}(z)$ is scalar and $\Sigma_{22}(z)$ is a (2×2) matrix. Let the chain scattering matrix $\vec{\Sigma}(z)$, if it exists, be defined as

$$\vec{\Sigma}(z)\begin{bmatrix}u_1(z)\\y_2(z)\\y_3(z)\end{bmatrix} = \begin{bmatrix}y_1(z)\\u_2(z)\\u_3(z)\end{bmatrix} \tag{2}$$

Lemma 1: The chain-scattering matrix, $\vec{\Sigma}(z)$ corresponding to the transfer matrix, $\Sigma(z)$ of the processing element exists if and only if $\Sigma_{22}(z)$ is nonsingular and $\vec{\Sigma}(z)$ is nonsingular if and only if, in addition, $\sigma_{11}(z)$ is non-zero. Furthermore, $\vec{\Sigma}(z)$ can be expressed either in the form

$$\vec{\Sigma}(z) = \begin{bmatrix}\frac{VU^T}{z-\beta} + \Theta\end{bmatrix} \tag{3a}$$

or in the form

$$\vec{\Sigma}(z) = \begin{bmatrix}zVU^T + \Theta\end{bmatrix} \tag{3b}$$

where $\{\beta, U^T, V, \Theta\}$ are constant matrices composed of finite elements and are of dimensions (1×1), (1×3), (3×1), (3×3) respectively.

Throughout this paper, it will be assumed that every processing element has a nonsingular chain-scattering matrix that can be represented in the form of Eq.(3a). Further it will be assumed that $VU^T \neq 0$. These assumptions are not particularly restrictive as will be shown in what follows.

Lemma 2: Let $\vec{\Sigma}(z)$ be the nonsingular chain scattering matrix of a processing element that can be represented in the form of Eq.(3a). Further, let

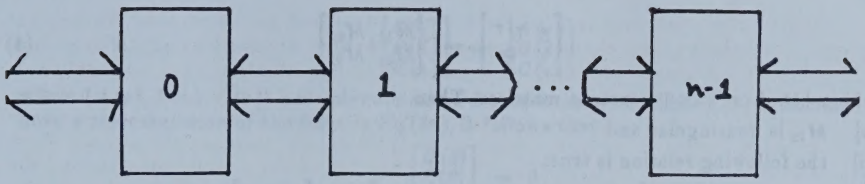


Fig.1. Locally Interconnected Array.

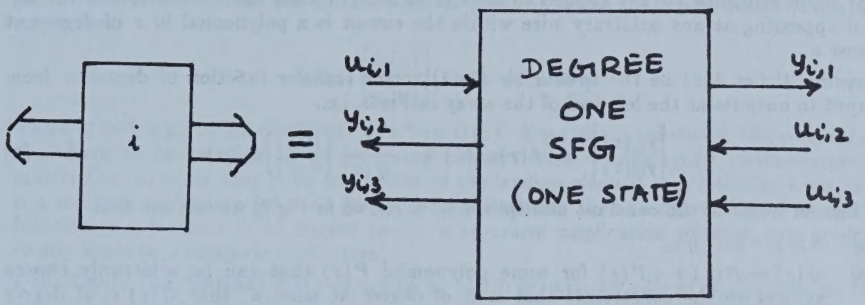


Fig.2. Imposing Modularity.

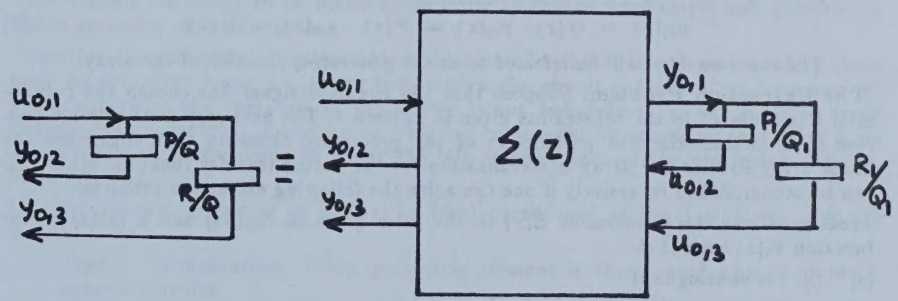


Fig. 4. The Extraction Problem.

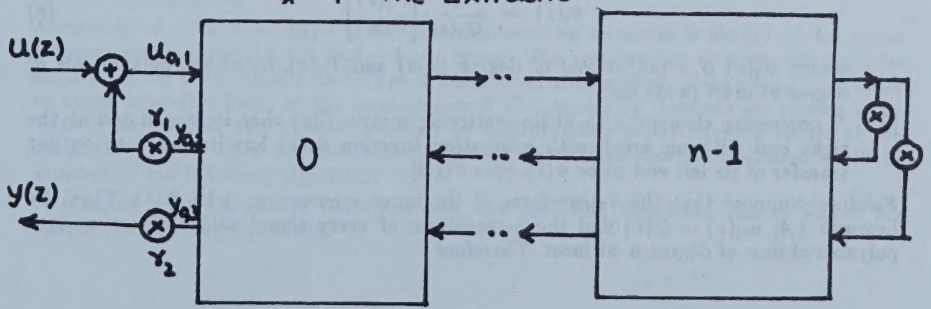


Fig.3. The SFG Realization.

$$\begin{bmatrix} \beta & U^T \\ V & \Theta \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{4}$$

where $\{M_{ij}\}$ are (2×2) constant matrices. Then

[a] M_{22} is nonsingular and

[b] the following relation is true:

$$\begin{bmatrix} M_{11} - M_{12}M_{22}^{-1}M_{21} & M_{12}M_{22}^{-1} \\ -M_{22}^{-1}M_{21} & M_{22}^{-1} \end{bmatrix} = \begin{bmatrix} a & B^T \\ C & D \end{bmatrix} \tag{5}$$

except for a normalization constant, where $\{a, B^T, C, D\}$ are as defined in Eq.(1).

Lemma 3: In any realization of $N(z)/D(z)$ with n state variables, if the z-transform of the input sequence $\{u(k)\}$ applied to the circuit is $D(z)$, then the z-transform of the signal appearing at any arbitrary wire within the circuit is a polynomial in z of degree at most n.

Lemma 4: Let $\Phi(z)$ be the irreducible (2×1) vector transfer function of degree n from input to outputs at the left end of the array in Fig.3, i.e.,

$$\begin{bmatrix} y_{01}(z) \\ y_{02}(z) \end{bmatrix} = \Phi(z)u_{01}(z) = \frac{1}{Q(z)}\begin{bmatrix} P(z) \\ R(z) \end{bmatrix} u_{01}(z) \tag{6}$$

Then, in terms of the constant multipliers γ_1, γ_2 shown in Fig.3, we can say that

[a] $R(z) = N(z)/\gamma_2$.

[b] $Q(z) = D(z) + \gamma_1 P(z)$ for some polynomial $P(z)$ that can be arbitrarily chosen subject to the constraints that it is of degree at most n, that $Q(z)$ is of degree equal to n and that $Q(z)$, $P(z)$, $R(z)$ are mutually coprime, and

[c] For input, $u(z) = D(z)$, the corresponding signals at the left end of the array are given by

$$u_{01}(z) = Q(z), \quad y_{02}(z) = P(z), \quad y_{03}(z) = R(z) \tag{7}$$

The function $\Phi(z)$ will be referred to as the *generating function* of the array.

The Extraction Problem: Suppose that the circuit designer has chosen the polynomial $P(z)$ subject to the constraints given in Lemma 4. The next step in the design process is to characterize the parameters of the processing elements that constitute the linear array so that the array is a realization for the vector transfer function $\Phi(z)$. This can be accomplished recursively if one can solve the following extraction problem:

Problem: Given $\Phi(z)$ determine $\vec{\Sigma}(z)$ in the form given in Eq.(3a) and a (2×1) vector function $\Phi_1(z)$ such that

[a] $\vec{\Sigma}(z)$ is nonsingular,

[b] $\Phi_1(z)$ is expressible in terms of polynomials as

$$\Phi_1(z) = \frac{1}{Q_1(z)}\begin{bmatrix} P_1(z) \\ R_1(z) \end{bmatrix} \tag{8}$$

where $Q_1(z)$ is a polynomial of degree $(n-1)$ and $P_1(z)$, $R_1(z)$ are polynomials of degree at most $(n-1)$ and

[c] A processing element with chain-scattering matrix $\vec{\Sigma}(z)$ that is terminated at the right end with an array with generating function $\Phi_1(z)$ has its input to output transfer at its left end to be $\Phi(z)$. (See Fig.4).

Solution: Suppose that the z-transform of the input sequence $u(z)$ be $D(z)$. Then, by Lemmas 1,4, $u_{01}(z) = Q(z)$ and the z-transform of every signal within the array is a polynomial in z of degree n at most. Therefore

$$\vec{\Sigma}(z)\begin{bmatrix} Q(z) \\ P(z) \\ R(z) \end{bmatrix} = \begin{bmatrix} A(z) \\ B(z) \\ C(z) \end{bmatrix} \tag{9}$$

where $\{A(z), B(z), C(z)\}$ are polynomials of degree n at most. Since $\vec{\Sigma}(z)$ is assumed to have a representation of the form in Eq.(3a), it follows that

$$U^T \begin{bmatrix} Q(\beta) \\ P(\beta) \\ R(\beta) \end{bmatrix} = 0 \tag{10}$$

From Eqs.(8,9), it is also evident that

$$A(z) = Q_1(z)p(z), \quad B(z) = P_1(z)p(z), \quad C(z) = R_1(z)p(z) \tag{11}$$

for some polynomial $p(z)$ of degree one at most. Now, if $p(z)$ is strictly of degree one, let $(z = \varsigma)$ be the zero of $p(z)$; otherwise let $(\varsigma = \infty)$. Then the relationships in Eq.(11) imply that

$$\vec{\Sigma}(z)\begin{bmatrix} Q(z) \\ P(z) \\ R(z) \end{bmatrix} = 0 \text{ at } z = \varsigma \tag{12}$$

where $Q(\infty)$, e.g., is the coefficient of z^n in $Q(z)$. Eqs.(10,12) constitute the constraints that have to be satisfied by a processing element with a nonsingular chain-scattering matrix $\vec{\Sigma}(z)$ in order that it be admissible as the leading element in a modular array that is a minimal realization for $\Phi(z)$. Its right neighbour can then be characterized using the function $\Phi_1(z)$ which is of degree $(n-1)$; a recursive application of such a procedure clearly leads to a complete realization.

Definitions: The constant β is defined to be the ***transmission pole*** of the processing element while ς is its ***transmission zero.***

In addition to the freedom in the choice of the polynomial $P(z)$, there is a considerable amount of freedom allowed to the designer in determining the parameters of $\vec{\Sigma}(z)$. This freedom turns out to be necessary in order to ensure pipelineable and numerically stable operation of the circuit.

Ensuring Computability: A computable realization of a digital filter is one in which there exist no delay-free loops, i.e., every loop within the circuit consists of at least one z^{-1} block; otherwise the operation of the circuit is not well-defined. In the modular array configuration considered in this paper, this can be ensured by arranging that there is no direct feedback transmission at either the left or right ends of the processing element. More formally,

Let the constant matrix Θ in Eq.(3a) be block-partitioned as $\{\Theta_{ij}\}$ with Θ_{22}, a (2×2) matrix. Then,

Type 1 Normalization: Every processing element is chosen such that Θ is block upper-triangular.

Type 2 Normalization: Every processing element is chosen such that Θ is block lower-triangular.

A cascade of Type 1 or Type 2 normalized processing elements is devoid of delay-free loops except possibly at the ends of the array. The terminating element can then be required to be both Type 1 and Type 2 normalized if necessary. Notice that it is easier to avoid delay-free loops at the terminations if the processing elements are chosen to be Type 2 normalized; this can be achieved by appropriately restricting the choice of the constant γ_1 or of $P(z)$. For this reason, Type 2 normalized processing elements will be assumed in the following algorithm.

Based upon the above considerations, the following algorithm can be stated for solving the extraction problem:

Algorithm:

Step 1: Choose a finite scalar constant β (transmission-pole) and a non-zero row vector U^T such that

$$U^T \begin{bmatrix} Q(\beta) \\ P(\beta) \\ R(\beta) \end{bmatrix} = 0 \tag{13}$$

Step 2: Determine the polynomial $W(z)$ of degree at most $(n-1)$ defined as

$$W(z) = \frac{1}{z-\beta} U^T \begin{bmatrix} Q(z) \\ P(z) \\ R(z) \end{bmatrix} \tag{14}$$

Step 3: Choose the transmission zero ς of the processing element. If the transmission zero is chosen to be finite, then it should be such that

$$W(\varsigma) \neq 0 \tag{15}$$

and the transmission zero can be chosen to be at the point at infinity provided $W(z)$ is strictly of degree $(n-1)$.

Step 4: Choose the constant matrix Θ to be block lower-triangular (for Type 2 normalization) such that Θ is nonsingular if the transmission zero is finite and

$$\Theta \begin{bmatrix} Q_0 \\ P_0 \\ R_0 \end{bmatrix} = 0 \tag{16}$$

if it is at the point at infinity.

Step 5: If ς is finite, then determine the column vector V as

$$V = \frac{-1}{W(\varsigma)} \Theta \begin{bmatrix} Q(\varsigma) \\ P(\varsigma) \\ R(\varsigma) \end{bmatrix} \tag{17}$$

Else if $\varsigma = \infty$ choose V to be any column-vector which has a non-zero (1,1) element.

Step 6: Determine the polynomials $Q_1(z)$, $P_1(z)$ and $R_1(z)$.

Remarks

[a] *Controllability:* Since the generating function of the linear array is assumed to be irreducible, it follows that the array is controllable from $\mathbf{u}_{0,1}(z)$. Now, $\mathbf{u}(z)$ is obtained from $\mathbf{u}_{0,1}(z)$ via state feedback. Using the invariance of controllability under state feedback, the realization is therefore controllable from $\mathbf{u}(z)$ even if the original description of the transfer function in terms of polynomials was reducible.

[b] *Dual Observable Realizations:* For every realization obtained using the algorithm, its dual realization can be derived by reversing the signal flow in every branch of the graph and transforming branch nodes into summers and summing nodes into branches. The dual realization is always observable.

[c] *Derivation of Existing Circuits:* Existing circuits that are configurable into a modular array of degree one sections can be derived using the above algorithm by following certain rules for making the various choices indicated in the algorithm. For example, the multi-rate systolic array in [21] is obtained by choosing

$$P(z) = z^n - D(z), \quad \gamma_1, \gamma_2 = 1$$

and the parameters of the Type 2 normalized processing element to be

$$\beta = 0, \quad \varsigma = \infty, \quad \Theta_{22} = I, \quad V^T, U^T = [1\,0\,0]$$

The remaining parameter Θ_{21} is obtained via Eq.(16). Note that the choice of U^T is admissible only because $Q(z) = z^n$ is zero at $z = 0$. Indeed, since $Q_1(z) = z^{n-1}$,

these rules can be applied for obtaining the other elements also.

[d] *Transmission and Reflection Type Realizations:* In general, the application of the above algorithm leads to a realization wherein the set of zeros of the transfer function from $\mathbf{u}(z)$ to $\mathbf{y}_{(n-1),1}(z)$, i.e., the set of transmission zeros of the processing elements, is quite different from the zeros of $H(z)$. However, it is conceivable that the designer deliberately chooses these transmission zeros to coincide with the zeros of $H(z)$ so that $\mathbf{y}_{(n-1),1}(z)$ is merely a scaled version of $\mathbf{y}(z)$; the array will be referred to as a *transmission type* realization in this case and to be a *reflection type* realization otherwise.

Pipelineability: The circuit that corresponds to the signal flow graph generated by the algorithm is pipelineable in operation if the transmission zero, ς, is chosen to be at the point at infinity at each iteration of the algorithm.

There are other conditions under which the circuit can be pipelineable but they turn out to be not as useful as the one above.

Pipelined Orthogonal Circuits: The chain-scattering matrix of a processing element with transmission zero at the point at infinity and which has an orthogonal transition matrix can be completely parametrized in terms of the (2×1) constant vector K as

$$\vec{\Sigma}(z) = \begin{bmatrix} z^{-1}(1-\bar{K}K)^{-1/2} & 0 \\ 0 & (I-K\bar{K})^{-1/2} \end{bmatrix} \begin{bmatrix} 1 & \bar{K} \\ K & I \end{bmatrix} \tag{18}$$

where $\bar{K}$ is the complex conjugate transpose of K and $\bar{K}K < 1$. Next, the constraints in Eqs.(13,16) necessitate the following conditions:

$$\Phi(\infty) = -K; \quad \bar{\Phi}(\infty)\Phi(0) = 1; \tag{19}$$

Clearly, such extractions are possible only for generating functions that have some special properties. Without going into details, it can be stated that if $\Phi(z)$ satisfies the following equations:

$$1-\bar{\Phi}(1/\bar{z})\Phi(z) = \frac{c}{\bar{Q}(1/\bar{z})Q(z)}, \quad c \geq 0 \tag{20}$$

and

$$|\Phi(z)| \leq 1 \quad \text{for} \, |z| \geq 1, \tag{21}$$

then every processing element can be chosen to have a chain-scattering matrix of the form in Eq.(18). This is substantiated in [9] wherein it is also shown that for any stable transfer function $H(z)$, there exist infinitely many choices of $\gamma_1, \gamma_2, P(z)$ for which $\Phi(z)$ satisfies (20,21). One special case has been derived independent of these considerations in [10].

REFERENCES

[1] L.Claesen et al., Proc. Europ. Solidstate Circ. Conf., Lausanne, Sept.83, pp.155-158.

[2] L.Claesen et al., Int. Journal of Cir. Theor. Appl., vol.11, pp.241-264, 1983.

[3] H. De Man et al., Second Int. Workshop on VLSI in Comm., IEEE Comm. Soc., June 1983.

[4] L. Gaszi et al., Proc. IEEE-Melecon, Athens, May 1983.

[5] M.Lowy et al., Internal Report, Dept. EE and CS., UC Berkeley, 1984.

[6] D.A.Schwartz, Proc. ICASSP, pp.8.3.1 - 8.3.4 Mar 1984.

[7] S.Y.Kung, Proc. IEEE, June 1984.

[8] H.V.Jagadish et al., Proc. Asilomar Conf., Jan. 1984.

[9] S.K.Rao and T.Kailath, Technical Report, Information Systems Lab., Stanford Univ., Feb. 1984.

[10] S.K.Rao and T.Kailath, IEEE Trans. CAS, Nov. 1984.

[11] P.Dewilde in Reviews of Modern Signal Processing, Hemisphere-Springer-Verlag, 1985.

[12] E.Deprettere et al., Int. Journal of Cir. Theor. Appl., 1980.

[13] D.Henrot and C.T.Mullis, Proc. Int. Conf. ASSP, pp.623-626, 1983.

[14] D.A.Schwartz and T.P.Barnwell, Proc. Int. Conf. ASSP, pp.30.13.1 - 30.13.4, Mar.1984.

[15] Mullis and Roberts, Proc. Int. Conf. ASSP, pp.8.8.1-8.8.4, Mar. 1984.

[16] H-H.Lu et al., Internal Memorandum, Dept. EE and CS., UC Berkeley, Aug.1984.

[17] Volder, J.E., IRE Trans. Electr. Comp., EC-8/3, pp.330-334, 1956.

[18] Ahmed et al., Computer, 15/1., pp.65-82, 1982.

[19] Delosme, Proc. Int. Conf. ASSP, 1983.

[20] E.F.Deprettere, Proc. Int. Conf. on ASSP, pp.41A.4.1 - 41A.6.4, Mar. 1984.

[21] S.Y.Kung in Reviews of Modern Signal Processing, Hemisphere-Springer-Verlag, 1985.

[22] S.K.Rao and T.Kailath, Internal Memorandum, Inf. Systems. Lab., Stanford Univ., 1983.

[23] T.Kailath, Linear Systems, Prentice-Hall, 1980.

[24] A.H.Gray and J.D.Markel, IEEE Trans. Acoust. Speech Sign. Proc., pp.486-494, Oct.1975.

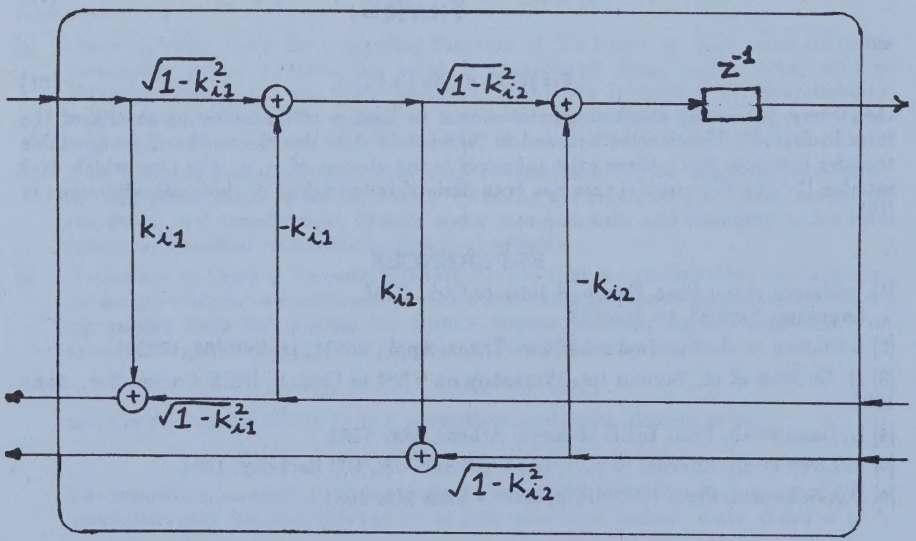


Fig. 5. An orthogonal Processing Element

PERFORMANCE EVALUATION OF A MULTIPLE ACCESS SYSTEM USING AN APPROXIMATE MODEL

RONGZHU ZHU AND ANTHONY EPHREMIDES

China Signal & Communication Company
Beijing, China

University of Maryland
College Park, MD 20742

ABSTRACT

An analysis using a coupled Markov Chain Model is developed for some variants of a slotted ALOHA system with a finite number of buffered nodes (in both symmetric and asymmetric). Analysis results of system delay under given transmitting probabilities as well as numerical optimization results have been obtained. Comparison between computational results and simulation for the various systems are provided. Agreement is excellent for most throughput values, but not for values near saturation. Numerical optimization results show that the asymmetric access scheme provides better delay-throughput performance than the symmetric access scheme for a two node system.

I. INTRODUCTION

The precise modeling of a slotted ALHOA system with a finite number of buffered nodes is straight forward by means of a M-dimensional infinite discrete random walk with appropriate boundaries. It is of course these boundaries that cause complications and make the analysis extremely difficult or impossible [1], [2]. Even approximate models are difficult to analyze [3], [4], [5]. So far the problem remains unsolved.

In this paper we develop an approximate model consisting of M+1 coupled Markov chains which decompose the above-mentioned M-dimensional infinite Markov chain into an M-dimensional finite system Markov chain and M infinite single-node Markov chains. As this method tracks the state of every node in every time slot, it not only enables to characterize the behaviors of the system under symmetric or asymmetric access, but also provides a better approximation than the one in [1] which is only for symmetric access system.

We use this model in three different slotted ALOHA systems. Node i receives packets according to an independent Bernoulli process with parameter λ_i. Node i attempts transmission over the common channel in three different ways.

In scheme I, the node consists of a buffer and a retransmitting unit. The departure probabilities of packet in node i are as follows:

$$P_r(\text{departure}) \triangleq \begin{cases} 1.0, \text{ ret. unit empty} \cap \text{buffer non-empty} & \text{(Active)} \\ 1.0, \text{ ret. unit empty} \cap \text{buffer empty} \cap \text{packet arrives} & \text{(Idle)} \\ P_i, \text{ ret. unit non-empty} & \text{(Blocked)} \end{cases} \tag{1}$$

In scheme II the node accesses the channel with a simpler scheme than in scheme I. A packet leaves the node if no other nodes transmit their packet, otherwise it returns to the head of the line (HOL) in

buffer. The departure probabilities of a packet in node i are as follows:

$$P_r(\text{departure}) \triangleq \begin{cases} P_i, \text{ buffer non-empty} & \text{(Non-empty)} \\ \\ P_i, \text{ buffer empty} \cap \text{packet arrives} & \text{(Empty)} \end{cases} \tag{2}$$

The difference between scheme III and scheme II is just such that if node is empty P_r(departure) of scheme III is one, e.i.

$$P_r(\text{departure}) \triangleq \begin{cases} P_i & \text{(Non-empty)} \\ \\ 1 & \text{(Empty)} \end{cases} \tag{3}$$

Scheme I coincides with the one considered in [1], while scheme II is the same as that of [2]. Scheme III is actually a intermedium of the others.

II. ANALYSIS OF A SLOTTED ALOHA SYSTEM WITH ACCESS SCHEME I

1. Modeling

In order to characterize the entire system behaviours, we must determine the probability distribution of the status of every node (active, idle, blocked) by the system Markov chain, and the probability distribution of queue length in every node in both blocked and unblocked (active and idle) states by the node Markov chain.

We denote the system state as $(S_1, \ldots S_i, \ldots S_M)$ for the system consisting of M nodes, where S_i is the state of node i. $S_i = 0$ if node i is idle, $S_i = 1$ if node i is active and $S_i = 2$ if node i is blocked. We define $P(S_1,\ldots S_i,\ldots S_M)$ to be the equilibrium probability of system state $(S_1,\ldots S_i,\ldots S_M)$.

In order to obtain the transition probabilities for system Markov chain, we introduce two conditional probabilities for node i

$$p(1/1)_i \triangleq P_r(\text{buffer content} = 1/\text{node is active}) \tag{4}$$

$$p(0/2)_i \triangleq P_r(\text{buffer content} = 0/\text{node is blocked}) \tag{5}$$

The expressions for $P(1/1)_i$ and $P(0/2)_i$ can be computed as shown in [6] along with the transition probabilities for the system Markov chain. If the quantitites of $P(1/1)_i$ and $P(0/2)_i$ are known, by solving the simultaneous equations for system Markov chain, the equilibrium probabilities of system state can be obtained.

The state of each node is characterized by the size of its buffer content, n_i, and by the indication S_i whether it is blocked or unblocked. Thus the state of node i is denoted by (S_i, n_i), where

$$S_i \triangleq \begin{cases} 0 & \text{node i is blocked} \\ 1 & \text{node i is unblocked} \end{cases} \tag{6}$$

Let $\pi_{S_i,n_i} \triangleq$ steady-state probability of state (S_i, n_i) and

$$G_{S_i}(z)_i \triangleq \sum_{n_i=0}^{\infty} \pi_{S_i,n_i} z^{n_i} \qquad S_i = 0,1 \tag{7}$$

In order to characterize the transition probabilities of the node Markov chain we need to define following conditional probabilities for node i:

$$P_{BS_i} \triangleq \Pr[\text{successful transmission/node i is blocked}] \tag{8}$$

$$P_{AC_i} \triangleq \Pr[\text{collision/node i is active}] \tag{9}$$

$$P_{IC_i} \triangleq \Pr[\text{collision/node i is idle}] \tag{10}$$

In terms of these quantities we can already characterize the transition probabilities of the node Markov chain as shown in detail in [6]. The quantities $\pi_{0,0}$, $\pi_{1,0}$, $\pi_{1,1}$, $G_0(1)$, $G_1(1)$ and N_i, the average number of packets in buffer, can be expressed in terms of λ_i, P_{SB_i}, P_{AC_i} and P_{IC_i}. Of course, the Markov chains defined here are only approximate models for the real system, since, as we know, the success probabilities in each slot depend on more than just the status of the individual node. Here we consider their <u>average</u> values as the <u>true</u> values.

If we consider first a two-node system, the success and collision probabilities as defined above are easily shown to be:

$$P_{BS_1} = P_1(P_{2,0}\overline{\lambda}_2 + P_{2,2}\overline{P}_2)/(P_{2,0} + P_{2,1} + P_{2,2}) \tag{11}$$

$$P_{BS_2} = P_2(P_{0,2}\overline{\lambda}_1 + P_{2,2}\overline{P}_1)/(P_{0,2} + P_{1,2} + P_{2,2}) \tag{12}$$

$$P_{AC_1} = 1 - (P_{1,0}\overline{\lambda}_2 + P_{1,2}\overline{P}_2)/(P_{1,0} + P_{1,2}) \tag{13}$$

$$P_{AC_2} = 1 - (P_{0,1}\overline{\lambda}_1 + P_{2,1}\overline{P}_1)/(P_{0,1} + P_{2,1}) \tag{14}$$

$$P_{IC_1} = 1 - (P_{0,0}\overline{\lambda}_2 + P_{0,2}\overline{P}_2)/(P_{0,0} + P_{0,1} + P_{0,2}) \tag{15}$$

$$P_{IC_2} = 1 - (P_{0,0}\overline{\lambda}_1 + P_{2,0}\overline{P}_1)/(P_{0,0} + P_{1,0} + P_{2,0}) \tag{16}$$

The total average delay incurred by a packet from the moment it is generated until it is successfully transmitted consists of three parts, waiting time in buffer, W_b, waiting time in retransmitting unit, W_r, and transmission time, T, (equal to one slot).

The average queue length in the buffer, N_i, can be determined by solving the equations of the M+1 coupled Markov chains and using characteristic function techniques as in [6]. It is given by

$$N_i = \lambda_i^2 \overline{\lambda}_i P_{IC_i} / [(\overline{\lambda}_i P_{BS_i} - \lambda_i P_{AC_i})(\overline{\lambda}_i P_{BS_i} - \lambda_i (P_{AC_i} - P_{IC_i}))] \quad (17)$$

The average queue length in the retransmitting unit is given by $G_0(1)$, namely

$$G_0(1)_i = \lambda_i \overline{\lambda}_i P_{IC_i} / [\overline{\lambda}_i P_{BS_i} - \lambda_i (P_{AC_i} - P_{IC_i})] \quad (18)$$

Using Little's formula [7] for node i, D_i is given by

$$D_i = W_{bi} + W_{ri} + T = (N_i + G_0(1)_i)/\lambda_i + 1 \quad (19)$$

and the total average system delay finally by

$$D = \sum_{i=1}^{M} (D_i \lambda_i) / \sum_{i=1}^{M} \lambda_i \quad (20)$$

2. Numerical Results

By applying an iterative procedure we obtain first the P_{BS_i}'s, P_{AC_i}'s and P_{IC_i}'s. Then, using (17), (18), we obtain D_i's and D from (19), (20).

For simplification, we assume $\lambda_1 = \lambda_2$ in the present example. We obtain the delay as a function of throughput for the symmetric case, where $P_1 = P_2$, and for the asymmetric case, where $P_1 \neq P_2$.

Figure 1 shows the numerically optimized delay vs. throughput for both the symmetric and the asymmetric cases.

III. ANALYSIS OF A SLOTTED ALOHA SYSTEM WITH ACCESS SCHEME II (or Scheme III)

We denote the system state as $(S_1,\ldots S_i,\ldots S_M)$, where S_i is the state of node i. $S_i=0$ if node i is empty, otherwise $S_i = \overline{0}$. $P(S_1,\ldots S_i,\ldots S_M)$ is the equilibrium probability of system state $(S_1,\ldots S_i,\ldots S_M)$.

We introduce the following conditional probability for node i

$$P(1/\overline{0})_i \triangleq \Pr(\text{buffer content = 1/node is non-empty}) \quad (21)$$

As shown in [6], this probability contributed crucially to the transition from $\overline{0}$ to 0 for node i.

Assuming the $P(1/\overline{0})_i$'s are known (for i = 1 to M), the transition probabilities can be obtained as shown in [6]. By solving the set of simultaneous equations for the system Markov chain, the equilibrium probabilities of that chain can be obtained.

The state of node i is only characterized by n, the queue length in the buffer of node i. Let π_{ni} be the equilibrium probability of state n for node i, and

$$P_{0S_i} \triangleq \Pr[\text{success/node is non-empty}] \tag{22}$$

$$P_{0C_i} \triangleq \Pr[\text{collision/node is empty}] \tag{23}$$

The equations for the node i Markov chain are as follows:

$$\pi_{0i} = \pi_{0i}(1-\lambda_i P_{0C_i}) + \pi_{1i}\lambda_i P_{\overline{0}S_i} \tag{24}$$

$$\pi_{1i} = \pi_{0i}\lambda_i P_{0C_i} + \pi_{1i}\left[1-\lambda_i(1-P_{\overline{0}S_i}) - \overline{\lambda}_i P_{\overline{0}S_i}\right] + \pi_{2i}\overline{\lambda}_i P_{\overline{0}S_i} \tag{25}$$

$$\pi_{ni} = \pi_{(n-1)i}\lambda_i(1-P_{\overline{0}S_i}) + \pi_{ni}\left[1-\lambda_i(1-P_{\overline{0}S_i}) - \lambda_i P_{\overline{0}S_i}\right] + \pi_{(n+1)i}\lambda_i P_{\overline{0}S_i} \quad n\geq 2 \tag{26}$$

By solving (24), (25), (26) for n = 2,3,...∞ as in [6], the steady-state probabilities π_{0i}, π_{1i}, $G_{\overline{0}}(1)_i = \sum_{n=1}^{\infty} \pi_{ni}$, and the average length of the queue in the buffer, $N_i = \sum_{n=1}^{\infty} (\pi_{ni} n)$, are obtained.

In scheme II and scheme III the total average delay in node i consists of the waiting time in the buffer and the transmission time T, namely

$$D_i = W_{bi} + T = N_i/\lambda_i + 1$$

$$= \overline{\lambda}_i P_{\overline{0}S_i} P_{0C_i} / [(P_{\overline{0}S_i} - \lambda_i)(P_{\overline{0}S_i} - \lambda_i(1-P_{0C_i}))] + 1 \tag{27}$$

Figure 2 shows the numerical result of delay vs. throughput for symmetric case of scheme II given $P_1 = P_2 = 0.5$.

Figure 3 shows the numerically optimized result of delay vs. throughput for symmetric case of schemes I, II, and III.

IV. COMPARISON

1. Comparison of delay vs. throughput between the numerical and the simulation results.

Figure 4 shows the comparison for the two node system under scheme I in the asymmetric case, with $P_1 = 0.4$, $P_2 = 0.6$.

Figure 5 shows the comparison for a three node system under scheme I in the asymmetric case, with $P_1 = 1.0$, $P_2 = 0.5$, $P_3 = 0.3$.

2. Comparison of delay vs. throughput between the numerical and the analytical results.

Figure 2 provides the comparison for two node system under scheme II in the symmetric case, where $P_1 = P_2 = 0.5$. The analytical result was derived by M. Sidi and A. Segall [2] and is the only one available in closed form.

3. Comparison of optimal delay vs. throughput between the numerical and the analytical results.

Figure 6 shows the comparison for a two node system under scheme II in the symmetric case. Analytical optimal delay in Figure 6 is derived from the formulae in [2].

4. Comparison of delay vs. throughput under numerical optimization between scheme I, scheme II and scheme III.

Figure 3 gives this comparison for a two node system in the symmetric case. Note that within the stable throughput region, the system optimal delay of scheme I is the best. This can be intuitively explained since the node under scheme I can adapt to the traffic on the channel by changing its transmitting probability between 1 and p_i.

5. Comparison of delay vs. throughput under numerical optimization between symmetric access and asymmetric access.

From Figure 1 we find that the optimal system average delay in the asymmetric case is distinctly better than that in the symmetric case for the two node system under scheme I. Numerical results show that the optimal transmitting probabilities in the asymmetric case are such that the transmission probability of one node is one (i.e. the node has full priority) constantly while that of the other node varies with the throughput.

V. CONCLUSION

1. In this paper we presented an approximate model for performance evaluation of slotted ALOHA systems with a finite number of buffered nodes. The model is not as inaccurate as that in [1] and thus a better approximation is obtained.

2. The model can be used in several variations of a slotted ALOHA system, allowing thus a comparison between the different possible access schemes. The comparison shows that system delay and capacity are the best for scheme I.

3. The model can be used in the symmetric as well as the asymmetric access scheme. The comparison between them shows that the optimal delay and capacity in the asymmetric system is better than that of the symmetric system. This property can be used to improve the performance of computer networks operating under certain contention condition.

4. The approach presented here can be utilized for the analysis of other contention-based protocol with buffered nodes.

REFERENCES

[1] Saadawi, T.N., and Ephremides, A., "Analysis, Stability, and Optimization of slotted ALOHA with a Finite Number of Buffered Users," IEEE Trans. on Automatic Control, Vol. AC-26, No. 3, pp. 680-689, June 1981.

[2] Sidi, M. and Segall, A., "Two Interferring Queues in Packet-Radio Networks", IEEE Trans. on Communications, Vol. 31, No. 1, pp. 123-129, January 1983.

[3] Kleinrock, L. and Yemini, Y., "Interferring Queueing Processes in Packet-Switched Broadcast Communication," IFIP Congress, Tokyo, 1980.

[4] Tobagi, F.A. and Kleinrock, L., "On the Analysis and Simulation of Buffered Packet Radio System, "Proceeding of the ninth Hawaii Int. Conf. on System Science, Honolulu, Hawaii Jan. 1976, pp. 42-45.

[5] Ephremides, A., "Analysis of Protocols of Multiple Access, "Lecture Notes on Modeling and Performance Evaluation, Springer Verlag, Proceeding of INRIA Symposium, Jan. 1983.

[6] Zhu, R., and Ephremides, A., "Performance Evaluation for Interacting Queues in Multiple Access Systems," System Research Report No., University of Maryland, August 1984.

[7] Little, J., "A Proof of the Queueing Formula $L = \lambda\omega$," Oper. Res., Vol. 9, pp. 383-387, Mar.-Apr. 1961.

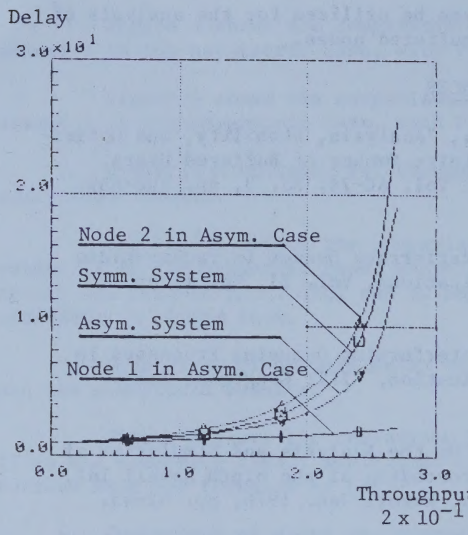


Figure 1. Optimal Delay vs. Throughput of Two Node System with Scheme I in Symmetric and Asymmetric Case.

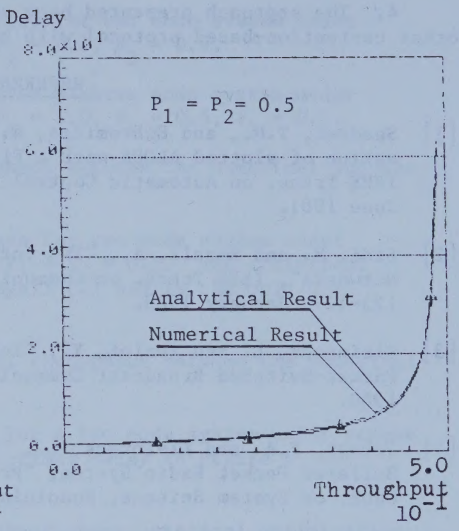


Figure 2. Delay vs. Throughput for Symm. Case of Scheme II

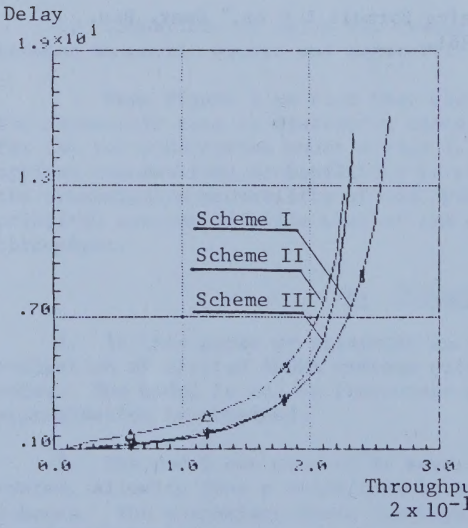


Figure 3. Delay vs. Throughput Under Numerical Optimization for Symm. Case.

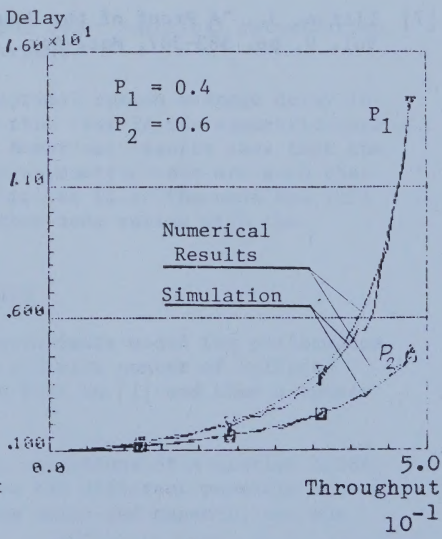


Figure 4. Comparison Between Numerical Results and Simulation (Two Node System)

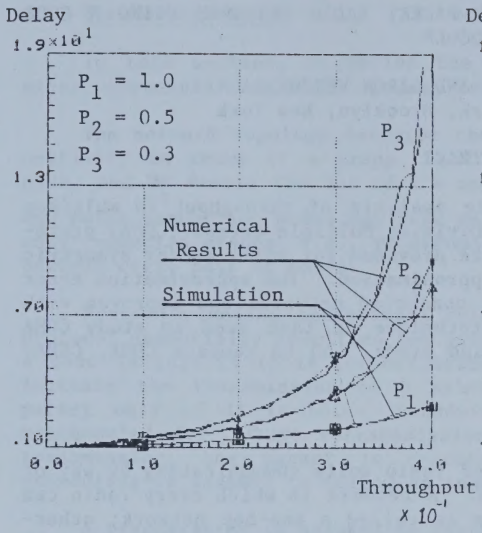


Figure 5. Comparison Between Numerical Results and Simulation (Three Node System).

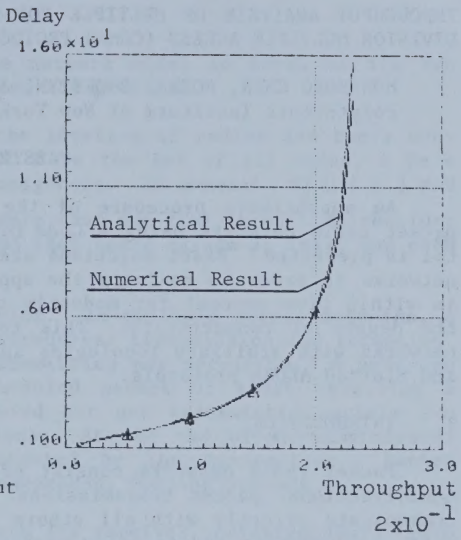


Figure 6. Comparison of Optimal Delay vs. Throughput Between Numerical and Analytical Results (Two Node System).

THROUGHPUT ANALYSIS OF MULTIPLE HOP PACKET RADIO NETWORKS USING A CODE DIVISION MULTIPLE ACCESS (CDMA) PROTOCOL*

MON-SONG CHEN, ROBERT BOORSTYN, AND AARON KERSHENBAUM
Polytechnic Institute of New York, Brooklyn, New York

ABSTRACT

An approximate procedure of the analysis of throughput in multihop packet radio networks using a Code Division Multiple Access (CDMA) protocol is presented. Exact solutions are provided for small and/or symmetric networks in order to evaluate the approximation. The approximation error is within five percent for modestly connected networks and improves with the degree of connectivity. This technique is then used to study CDMA networks with arbitrary topologies and sizes, and to compare CDMA, CSMA, and Slotted ALOHA protocols.

1. INTRODUCTION

Packet radio networks consist of radio units communicating by way of omnidirectional packet transmissions. A network in which every radio can communicate directly with all others is called a one-hop network; otherwise, intermediate radios relay packets, thus creating a multihop network.

Various multiple access schemes may be used in packet radio networks to achieve resource sharing efficiency depending on the traffic profile, network connectivity, and performance objectives. For one-hop networks, several protocols have been proposed and thoroughly studied. The main issue in these protocols is how to coordinate packet scheduling in time, to avoid collisions and improve throughput. If these protocols are applied to multihop networks, however, timing and spatial coordination jointly determine their performance; this makes the analysis significantly more complex. Nevertheless, a number of performance analyses of multihop packet radio networks have been proposed, and a brief review follows.

In [1], Tobagi analyzed a two-hop centralized Carrier Sense Multiple Access (CSMA) network. Silvester studied randomly dispersed multihop packet radio networks under Slotted ALOHA and concluded in [2] that "Six is a Magic Number." In [3-5], Boorstyn et al. introduced a Markov process model for multihop packet radio networks using CSMA. This model led to an automated procedure capable of analyzing networks of realistic size and arbitrary topology [6]. An extension of the Markov model to Busy Tone Multiple Access (BTMA) was presented in [7]. Analyses of simplified versions of unslotted ALOHA and unslotted CDMA, were proposed in [8].

In this paper, an approximation for the analysis of throughput of multihop packet radio networks using a CDMA protocol is developed. Approximate solutions are compared with the exact solutions for some small and/or symmetric networks. The approximation error is found to be within five percent for modestly connected networks and improves with the degree of connectivity. This technique is then used to study CDMA networks with arbitrary topology and size, and to compare CDMA, CSMA, and Slotted ALOHA protocols.

---

* This research was partially supported by DARPA under contract MDA903-82-C-0457.

2. The Network Model

In this section, we define the network model in terms of its two major components: network topology and the protocol.

The network topology describe the location of radios and their connectivity in terms of a graph. Let V be the set of all nodes, i be a node, and N_i^* denote the set of i's neighbors. In general, $N_i^* \cup \{ i \} \neq V$ and paths connecting nodes may have more than one link (hop). We consider only connected graphs, i.e., we assume that there exists at least one path between every pair of nodes.

Protocols define operational rules. In multiple access networks, a protocol essentially specifies two procedures: transmission and reception. A node is idle if it is neither transmitting nor receiving. A node can initiate the transmission of a scheduled packet or start receiving a packet only if it is idle. Scheduled but not transmitted packets are rescheduled for future retransmissions. At the end of an unsuccessful transmission, the packet is rescheduled by the transmitter. Packet scheduling is assumed to follow an independent Poisson process.

A transmission is assumed to reach its receiver instantaneously (zero propagation delay between neighbors). If the addressed receiver is idle at the initiation of a packet transmission, it starts receiving the packet successfully. Once a successful reception is initiated, the receiver locks on to it by a code division mechanism, thus the transmission remains immune throughout the packet duration (perfect capture). Finally, packet lengths may have arbitrary distributions.

An idle node may receive a packet even if some of its neighbors are transmitting. This implies the use of orthogonal codes. Code Division Multiple Access (CDMA) techniques use spread spectrum waveforms such as direct sequence pseudo-noise modulation, frequency hopped modulation, and hybrid combinations of these two [9]. Transmissions coded in this way provide receivers with the ability to receive one transmission and reject others as noise. Note that this selective reception ability may be constrained by signal-to- noise ratio (SNR) thresholds which place a limit on the number of simultaneous successful transmissions in a neighborhood. Our model assumes that nodes of CDMA networks always have the selective reception ability, i.e., SNR thresholds are never violated. This is a reasonable assumption when nodes are not too heavily clustered. Subsequent work has included this noise phenomenon.

3. The CDMA Approximate Analysis

In this section, we develop an approximation for the network model under consideration. Let i be a node, and N_i^* the set of its neighbors. Let g_{ij} be the rate (in packets/sec) of packets scheduled at i and destined to a neighbor j. This includes original and rescheduled packets and is assumed to be Poisson. Let s_{ij} be the required (successful) traffic rate (in packets/sec) from node i to node j, and $1/\mu_{ij}$ be the average length of packets transmitted from i to j. The s_{ij} are determined from the end-to-end requirements and the routing which is assumed to be known.

We denote by P_i, T_i and R_i the probabilities that at a random instant i is idle, transmitting and receiving, respectively. These steady state probabilities also denote the normalized proportions of time a node is in a particular state. Since node i is either transmitting, receiving, or idle, we have

$$P_i + T_i + R_i = 1 \tag{1}$$

Packets are scheduled according to an independent Poisson process, and only those that find i idle are transmitted. Thus

$$T_i = \left(\sum_{j \varepsilon N_i^*} g_{ij}/\mu_{ij} \right) P_i \tag{2}$$

Due to the perfect capture assumption, the time a node is receiving is devoted entirely to successful packets. Hence, the probability that i is receiving is proportional to the total duration of successful packet transmissions destined to i. Therefore

$$R_i = \sum_{j \varepsilon N_i^*} s_{ji}/\mu_{ji} \tag{3}$$

Let $G_{ij} = g_{ij}/\mu_{ij}$ and $S_{ij} = s_{ij}/\mu_{ij}$ be the normalized rates, we then have the following

$$P_i = \frac{1 - R_i}{1 + \sum_{j \varepsilon N_i^*} G_{ij}} \tag{4}$$

where

$$R_i = \sum_{j \varepsilon N_i^*} S_{ji}$$

A successful i to j transmission requires three conditions: (1) a packet is scheduled at i for j, (2) i is idle and (3) j is idle. Thus

$$\begin{aligned} S_{ij} &= G_{ij} \cdot \text{Prob (i and j are idle)} \\ &= G_{ij} \cdot \text{Prob (i is idle)} \cdot \text{Prob (j is idle/i is idle)} \end{aligned} \tag{5}$$

To complete the analysis, we need the joint probability that both i and j are idle or the conditional probability that j is idle given that i is idle. An exact approach is to solve for the steady state probabilities of the underlying continuous time Markov process (if we assume exponential packet lengths). This would be practical for a network of non-trivial size, if the process had product form state probabilities as in the CSMA models of [3-6]. In CDMA, however, the process does not fall under the class of reversible processes that lead to simple product form evaluations. We can easily demonstrate this by showing that Kolmogorov's criteria for reversibility, [10], are violated. For example, consider a state S whereby one node is transmitting unsuccessfully while all others are idle. This is possible when a node is unsuccessfully transmitting to a receiver that was initially busy but since switched to idle. If the transmitter in state S finishes its transmission before any other event occurs, the system switches from state S to a state ϕ in which all nodes are idle. However, there is no legitimate transition from state ϕ to

state S. Unilateral transitions of this kind indicate that reversibility does not hold, thus the tractable product form algorithms in [3-6] do not apply. Hence, we are motivated to find an alternative approach -- an approximation to handle realistic size networks.

Consider a common time interval (normalized to unity) and the activities of nodes i and j, as shown in Figure 1. $(\bar{T}_i + S_{ij})$ and $(\bar{R}_i + S_{ji})$ are the time periods during which i transmits and receives, respectively. The minus sign indicate that S_{ij} and S_{ji} have been removed during the intervals S_{ij} and S_{ji}, successful transmissions from i to j or from j to i are in progesss, and both nodes are aware of this activity. $\bar{T}_i$ and $\bar{R}_i$ are split into the three regions shown. Outside S_{ij} and S_{ji}, almost every combination of status of i and j is possible, except successful transmissions that are to/from i and j and both involve one of their common neighbors. In other words, i and j do not influence each other directly, and only some indirect dependence is created through their common neighbors. Nevertheless, because of the large number of possible joint states and relatively small restrictions, we believe that the indirect dependence is weak enough to be neglected, and we will assume that nodes i and j are independent outside intervals S_{ij} and S_{ji}.

Transmission from i to j are initiated only when i is idle; during this interval i is not successfully transmitting to or receiving from j. Hence, from j's viewpoint, transmissions from i to j are initiated in an interval $(1 - S_{ij} - S_{ji})$ on the average. Assuming that nodes i and j are independent during $(1 - S_{ij} - S_{ji})$, the conditional probability that j is idle given that i is idle is the ratio of j's normalized idle time period to the interval $(1 - S_{ij} - S_{ji})$. Thus

$$\text{Prob (j is idle/i is idle)} \cong \frac{P_j}{1 - S_{ij} - S_{ji}} \tag{6}$$

Hence

$$S_{ij} \cong G_{ij} \cdot P_i \cdot \frac{P_j}{1 - S_{ij} - S_{ji}} \tag{7}$$

Similarly, let t_{ij} be the transmission rate (in packets/sec) from i to j and $T_{ij} = t_{ij}/\mu_{ij}$ be normalized rate, we have

$$t_{ij} = g_{ij} \cdot P_i$$

and

$$T_{ij} = G_{ij} \cdot P_i$$

Since $P_j = 1 - T_j - R_j$, Eq. (7) is equivalent to

$$S_{ij} \cong T_{ij} \frac{1 - R_j - T_j}{1 - S_{ij} - S_{ji}} \tag{8}$$

Based on Eq.s (7) or (8), throughputs of CDMA networks can be approximately analyzed using an iterative procedure. Based on the end-to-end traffic requirements and the network routing, we can determine the

desired link rates S_{ij}. A set $\{S_{ij}\}$ is feasible if there exists a supporting set $\{G_{ij}\}$ and $\{T_{ij}\}$. To find $\{G_{ij}\}$ or $\{T_{ij}\}$, we iterate in the following manner: at the kth iteration

$$G_{ij}^{(k)} = \frac{S_{ij}\,(1 - S_{ij} - S_{ji})}{P_i^{(k-1)} \cdot P_j^{(k-1)}}$$

$$P_i^{(k-1)} = \frac{1 - R_i}{1 + \sum_{j \varepsilon N_i^*} G_{ij}^{(k-1)}}$$

$$T_{ij}^{(k)} = \frac{S_{ij}\,(1 - S_{ij} - S_{ji})}{1 - R_j - T_j^{(k-1)}}$$

where

$$G_{ij}^{(0)} = S_{ij}$$

$$T_{ij}^{(0)} = S_{ij}$$

Our experience has been that if the $\{S_{ij}\}$ is feasible, the iteration converges. To determine the maximum throughput, we scale every S_{ij} by a common factor; the maximum throughput corresponds to the largest scale factor that yields a feasible $\{S_{ij}\}$.

In order to evaluate the approximation, some exact solutions need to be found. Although general exact analyses are computationally prohibitive, there are a few small and/or symmetric configurations that can be analyzed. We developed exact procedures for uniform link traffic on small rings, chains, star networks with one node on each leg, fully connected (one hop) networks, a 3 by 3 closed grid, and a four node five link network, as shown in Figure 2. Table 1 summarizes the results. The approximation exhibits larger errors (about 10%) in networks that have minimal connectivity like chains and rings, and it is within 5% accuracy for modestly connected networks. We believe that the higher the connectivity, the better the results of the approximation.

4. Comparison of Protocols

In this section, we compare CDMA, CSMA, and Slotted ALOHA in various network topologies. Eq. (7) (for CDMA), the SP algorithm in [6] (for CSMA), and the model in [2] (for Slotted ALOHA) are used to compute maximum throughputs. The networks considered include large and uniform networks, random networks, and some specific topologies (chains, rings, stars, etc.).

A. Large Uniform Networks:

We considered large closed uniform networks in which every node and link are topologically identical. Examples considered are rings, triangular grids, rectangular grids, and hexagonal grids. For these networks, uniform end-to-end requirements result in uniform link and nodal traffics; hence, the network throughput γ can be evaluated by

$$\gamma = \frac{n \cdot S_n}{PL}$$

where S_n is the nodal throughput, PL the average path length, and n the number of nodes of the network.

In these planar uniform networks, each node has h·d nodes that are h hops away from it, where d is the degree of the nodes. In this case, PL can be approximated by

$$PL \cong \frac{2}{3}\sqrt{\frac{2}{d}}\sqrt{n}$$

Hence

$$\gamma \cong \frac{3}{2}\sqrt{\frac{d}{2}}\, S_n \sqrt{n} \qquad (9)$$

To find S_n, analytical expressions are available for CDMA and Slotted ALOHA, while the numerical procedure in [6] is needed for CSMA. The expressions for maximum nodal throughputs are

$$S_n = \frac{3 - 2\sqrt{2}\sqrt{1 - 1/d}}{1 + 8/d} \qquad (10)$$

for CDMA and

$$S_n = \frac{d^d}{(d+1)^{d+1}} \qquad (11)$$

for Slotted ALOHA. As for CSMA, Table 2 shows some throughputs for closed grids, which eliminate the edge effect and approximate large uniform networks. These results asymptotically project throughput for networks of considerably larger size.

Table 3 summarizes the comparisons. CDMA is significantly better than the others, while CSMA is either better than or about the same as Slotted ALOHA. For all protocols, ring connected networks have fixed throughput, while network throughput for degrees of 3, 4, and 6 grow with the square root of the number of nodes n. Note that CDMA networks achieve maximum throughput of .17 * n when fully connected. This is caused by the assumption that nodes always have the selective reception ability, thus higher connectivity does not limit in any sense the network throughput. When the noise effect is introduced, the throughput will be drastically reduced.

B. Random Networks:

In this comparison, networks are generated by placing nodes randomly and connecting them according to their Euclidean distances. Uniform link traffic is imposed for simplicity of comparisons. The results are provided in Table 4. CDMA is about 2 - 3.5 times better than CSMA, while CSMA is 1.5 - 2 times better than Slotted ALOHA.

C. Specific Networks:

Comparisons for rings, chains, rectangular grids, and star networks are listed in Tables 5 and 6, again under uniform link traffic. CDMA

performs better than CSMA and Slotted ALOHA, except in small chains and rings. CSMA is either better than or about the same as Slotted ALOHA.

5. Conclusions

In this paper, we have developed and evaluated an approximation for the analysis of throughput in multihop packet radio networks that use a CDMA protocol. The approximation is validated by comparing it with the exact analysis for some small and/or symmetric networks. The accuracy of the approximation is within five percent for modestly connected networks and improves with connectivity. We also compare CDMA, CSMA, and Slotted ALOHA in various network topologies. In general, CDMA is significantly better than CSMA and Slotted ALOHA, while CSMA is either better than or about the same as Slotted ALOHA.

Although the approximation is quite satisfactory, the model has one drawback, namely the assumption that nodes always have the selective reception ability. This may be unreasonable in highly connected networks. As a result of the assumption, fully connected networks achieve maximum network throughputs; furthermore, in fully connected networks, the throughput increases linearly with the number of nodes. We are currently extending our model to include the effect of SNR that limits the capture capability of receivers.

Finally, it is worthwhile to point out that the approximation is insensitive to two assumptions in the network model, namely the exponential packet length and the zero propagation delay. Firstly, since the approximation uses only the average packet lengths in computing various probabilities, it is insensitive to the packet length distribution. Secondly, since we assumed independence between nodes i and j outside intervals S_{ij} and S_{ji}, the time lapse due to propagation delays is hardly a factor.

REFERENCES

1. Tobagi, F. A., "Analysis of a Two-Hop Centralized Packet Radio Network Part II: CSMA," IEEE Trans. on Communications, Vol. COM-28, Feb. 1980.

2. Kleinrock, L. and J. Silvester, "Optimum Transmission Radii for Packet Radio Networks, or Why Six is a Magic Number," Proc. of the NTC-78, Birmingham, Al., Dec. 1978.

3. Boorstyn, R. and A. Kershenbaum, "Throughput Analysis of Multihop Packet Radio," Proc. of the ICC-80, Seattle, WA., June 1980.

4. Boorstyn, R., A. Kershenbaum, and V. Sahin, "A New Acknowledgement Protocol for Analysis of Multihop Packet Radio Networks," Proc. of the COMPCON-82, Washington, D.C., Sep. 1982.

5. Maglaris, B., R. Boorstyn, and A. Kershenbaum, "Extension to the Analysis of Multihop Packet Radio Networks," Proc. of the INFOCOM-83, San Diego, CA, April 1983.

6. Kershenbaum, A. and R. Boorstyn, "Evaluation of Throughput in Multihop Packet Radio Networks with Complex Topologies," Proc. of the INFOCOM-83, San Francisco, CA., April 1984.

7. "Analysis of Busy-Tone Multiple Access Protocols for Multi-Hop Packet Radio Networks," W. C. Chen, Ph.D. Dissertation, Polytechnic Institute of New York, June 1983.

8. Tobagi, F. A. and J. M. Brazio, "Throughput Analysis of Multihop Packet Radio Networks under Various Channel Access Schemes," Proc. of the INFOCOM-83, San Diego, CA., April 1983.

9. Kahn, R. E., S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman, "Advances in Packet Radio Technology," IEEE Proceedings, VOL. 66, No. 11, Nov. 1978.

10. "Reversibility and Stochastic Networks," F. P. Kelly, Wiley, NY, 1979.

Table 1. Comparison of maximum link throughput for exact solution and approximation.

			MAXIMUM LINK THROUGHPUT	
Network type	Network size	Exact Solution	Approximation	Difference (%)
Ring	3	.105	.100	5
	4	.113	.100	12
	5	.110	.100	9
	6	.109*	.100	8
	7	.109*	.100	8
Chain	3	.170	.146	14
	4	.134	.121	10
Star	3**	.116	.104	10
	4**	.088	.082	7
	5**	.074	.068	8
	6**	.060	.058	3
	7**	.053	.051	4
	8**	.047	.046	2
Fully Connected	3	.105	.100	5
	4	.065	.063	3
	5	.047	.046	2
	6	.037	.036	3
	7	.031	.030	3
	8	.026	.025	3
Closed Grid	9	.048*	.046	4
Asymmetric	4	.077	.073	5

* Simulation. **Number of legs.

Table 2. Throughputs of closed uniform CSMA networks.

Degree	Node Number	Maximum Nodal Throughput
3	8	.0987
	12	.104
	16	.104
	20	.104
4	9	.0928
	16	.0760
	25	.0804
6	16	.0674
	20	.0674

Table 3. Network throughputs in large uniform networks.

	NETWORK THROUGHPUT		
Degree	CDMA	CSMA	Slotted ALOHA
2	.80	.69	.59
3	$.35\sqrt{n}$	$.19\sqrt{n}$	$.19\sqrt{n}$
4	$.39\sqrt{n}$	$.17\sqrt{n}$	$.17\sqrt{n}$
6	$.47\sqrt{n}$	$.18\sqrt{n}$	$.15\sqrt{n}$
n-1	.17n	1	$\frac{1}{e}$

Table 4. Comparison of Protocols for Random Networks

Number of Nodes	Degree of Connectivity	Link Throughput CDMA	CSMA	Slotted ALOHA
20	10.6	.0139	.0039	.0024
20	10.0	.0141	.0039	.0023
18	4.6	.0278	.0141	.0090
20	7.4	.0194	.0069	.0048

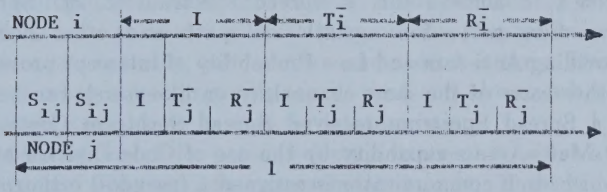
Table 5. Comparison of Protocols fo Star Networks (equal link requirements)

Five nodes in each leg	CDMA (approx.)	CSMA	Slotted ALOHA
Number of legs	Throughput of center node, Snode		
2 (∞ nodes/leg)	.200	.172	.148
3	.254	.207	.189
4	.284	.216	.205
5	.304	.220	.212
	Successful link rates, S_{ij}		
2 (∞ nodes/leg)	.100	.086	.074
3	.085	.069	.063
4	.071	.054	.051
5	.061	.044	.042
Star with five legs			
Nodes in each leg			
1	.074	.058	.055
2	.061	.046	.044
3	.061	.044	.043
4	.061	.044	.043

Table 6. Comparison of Protocols for Some Specific Networks (equal link requirements)

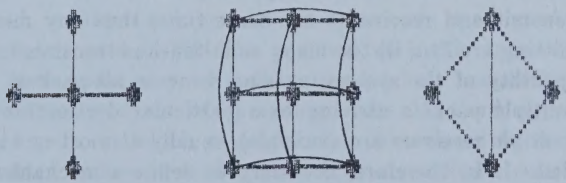
Topology	Number of nodes	Successful Link Rates CDMA	CSMA	Slotted ALOHA
Chain	3	.170*	.167	.135
	4	.134*	.128	.106
	5	.111	.111	.094
	6	.107	.102	.088
	7	.104	.097	.084
	8	.103	.094	.081
	∞	.100	.086	.074
Ring	3	.105*	.167	.074
	4	.113*	.073	.074
	5	.111*	.100	.074
	6	.100	.083	.074
	7	.100	.087	.074
	8	.100	.085	.074
	∞	.100	.086	.074
Open Grid	4	.100	.073	.074
	9	.062	.036	.038
	16	.053	.028	.029
	25	.050	.024	.025

*exact results



(I indicates idle)

Figure 1 Activities of i and j in a common time interval



a.Star Network b. 3 by 3 Closed Grid c. 4 Node 5 Link Network

Figure 2 Sample networks for which exact throughputs are available

PERFORMANCE OF SPREAD SPECTRUM NETWORKS

JOHN A. SILVESTER
Communications Sciences Institute, USC
Los Angeles, CA 90089-0781

ABSTRACT

In this paper we describe throughput/delay performance results for Spread Spectrum Communication Networks. In previous work we have used a strong independence assumption and an iterative algorithm to generate approximate performance models for (non-spread) multiple access broadcast networks using random access protocols. The model incorporates the effect of buffering and finite populations. In this paper we extend these results to Spread Spectrum Networks capable of Code Division Multiple Access and study the effect of the multi-user threshold and capture.

INTRODUCTION

In this paper we study the performance of Spread Spectrum Packet Radio Networks. Packet Radio Communication Networks are an attractive solution for supporting communication between a potentially mobile set of bursty users [1]. Although Packet Radio has found application in civilian life, its major utilisation has been for military networks. In the military domain it is usually necessary to protect radio communication against hostile (as well as environmental) interference and a common approach to achieve this is Spread Spectrum [2, 3]. Spread Spectrum communication offers protection against multipath and other noise sources, in addition to providing Anti-Jam and Low Probability of Intercept protection.

Since other users of the same channel, (even like-users), can be made to look like noise to a Spread Spectrum receiver, Spread Spectrum Communication also offers inherent Multi-Access capability by the use of Code Division Multiple Access (CDMA), in which each communicator is assigned a (pseudo-) orthogonal code to be used for transmissions.

Even with this multi-access capability, the possibility of collisions (unsucessful transmissions) still exists, since: i) it is typically the case that a node cannot simultaneously transmit and receive at the same time, thus any messages arriving while he is transmitting are lost; ii) too many simultaneous transmissions may exceed the multi-user capability of the system, causing some or all packets to be received incorrectly; iii) multiple packets arriving to a particular destination cannot all be received (unless multiple receivers are available), usually at most one of them can be successfully decoded. It is, therefore, necessary to define a mechanism for deciding when to use the channel - *the access protocol*, which may (or may not) be similar to the protocols used for non-Spread Spectrum Radio communication.

This work is supported in part by the Naval Air Systems Command (NASC) under contract No. N00019-83-C-0349. and in part by the Army Research Office (ARO) under contract No. DAAG29-84-K-0084.

We see, therefore, that although the collision problems have been reduced, the scenario is very similar to the standard broadcast networks that have been previously studied. In particular, we can anticipate that the underlying queueing processes will be complex and not lend themselves to exact solution. Previous work on delay throughput for Spread Spectrum networks has concentrated on specific topologies and not allowed for the possibility of buffering. Davis and Gronemyer, [4], were perhaps the first to study CDMA and looked at a centralised network and studied the effect of capture (achieved by randomly staggering the transmission times of the packets). Another study by Raychaudhuri [5] studied a network of paired-off users, (i.e., each node had a specific partner with wich he communicated), and studied the multi-user capability.

In this paper we present extensions to some earlier work [6, 7], in which we developed an approximate model for the delay/throughput performance of Slotted Aloha, [8], non-Spread Spectrum Radio Networks. In this paper, these results are extended to Spread Spectrum and some initial results for simple network configurations are presented. These results are compared to simulation. The impact of the multi-user threshold and the benefits of the capture effect are also investigated.

NETWORK MODEL

The main differences between Spread Spectrum and non-Spread Spectrum Networks are: i) whereas for a non-Spread System it is typically assumed that at most one message can be successfully transmitted at any time, Spread Spectrum readily allows multiple successful simultaneous transmissions; ii) in a non-Spread System, a node is able to monitor all activity on the channel, whereas a Spread Spectrum receiver can typically only monitor one code at a time. This second point has an impact on what access protocols can be used and also on what protocols can be used for selecting the code to be used for a transmission.

The simplest approach for a code selection protocol is to use a *common-code* for all transmissions. This was the approach taken in [4]. If the transmissions are staggered in time it is still possible to capture one of the transmissions due to the phase difference in the code. If, however, the transmissions are synchronized, the system reduces to a non-spread Aloha type system. Alternately a *transmitter-based* code can be used, in which a unique transmitting code is assigned to each terminal, this is the approach used in [5]. The problem with this approach is that an idle terminal must have some way of anticipating packets transmitted to itself, i.e. a protocol for monitoring a set of spreading codes. A third approach is to use a *receiver-directed* code, in which a unique code is assigned to each receiver. Thus when a transmitter wishes to send a packet to a particular receiver, it selects the code specific to that receiver. The code thus acts like an address. Various hybrid combinations of above basic shemes are, of course, possible and in [9, 10] we investigate two such variations. In this paper we restrict our attention to receiver directed codes although the same approach can be applied to other protocols.

GENERAL ANALYTICAL APPROACH

The main problem with studying the delay performance of buffered broadcast networks, is that the service time at node i depends on whether the destination node is busy or not and on the current channel activity due to (all) other nodes. This makes the underlying queueing processes difficult to solve analytically [11].

Segall and Sidi [12] overcame this problem by assuming that nodes are always busy, this leads to pessimistic results. Saadawi and Ephremides [13], modelled the network at two levels, the system level, which describes which how many nodes are busy, and the node level, which describes the queueing behaviour at a node in terms of the system state. Although this model was very useful for a fully-connected network, it does not readily extend to the multi-hop case. Our approach, which is a combination of these two approaches, assumes independence between the queues and parameterizes the interaction with a steady-state estimate of the state of the other nodes. We first applied this to a Multi-Hop Slotted Aloha Network [6, 7] and have also studied other access protocols. Since this results in an approximate solution, the results are validated using simulation.

General Assumptions

We consider a network of n packet radio units, which can act as a traffic source and as a repeater for multi-hop traffic. The access protocol is receiver-directed Slotted Aloha. Node i transmits the message at the head of its queue, (when busy), with probability p_i in every slot until it is successful, (which it determines by a instantaneous free acknowledgement). This model, therefore, corresponds to the 'Delay first Transmission' (DFT) protocol and is considered here mostly for reasons of the simplicity of the resulting expressions. In a real network is it usual to transmit a packet with probability 1 for the first transmission and only use the (re)-transmission probability p for retransmissions, if necessary.

For a transmission to be successful, the following conditions must hold: i) less than some threshold number of users transmits, this is the so-called multi-user threshold above which the noise due to the other users causes the number of errors to exceed the error correcting capability of the codes used (this is, of course, not really a threshold rather a smooth degradation, but using a threshold model appears to be a reasonable first approximation, however); ii) the destination is not transmitting, since his transmitted power would swamp his own receiver, (it may be possible to transmit and receive at the same time for certain systems); iii) if multiple packets are addressed to the same destination, then the first packet has a sufficient phase offset to allow it to be 'captured', it is typically assumed that the other packets are lost.

Parameters of the Model

In order to proceed with the analysis we define the following parameters:

a. γ_{sd} : this is the externally generated end-to-end traffic from s to d;

b. $r_i = \sum_d \gamma_{id}$: external arrivals at i, summed over all destinations, this is assumed to be Bernouilli.

c. s_i : the traffic that passes through i on its way to its final destination, it is found from { γsd } and the routing algorithm;

d. λ_i : (attempted) arrival rate of transit traffic at i

e. β_i : the probability that a transmission from i is successful, i.e. there is no (destructive) interference;

f. b_i : the probability that node i is busy, it is through this parameter that the interaction with the other nodes is modelled, it is not known a priori, but is itself determined as part of the solution.

The Associated Markov Chain

In Figure 1, we show the Markov Chain for the queue size at some node in the network.

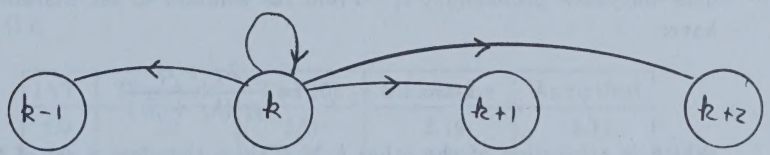


Figure 1: Markov Chain for a Generic Node

The associated transition probabilities are given by:

$$\begin{aligned} p_{kk-1} &= p\,\beta(1-r) \qquad (= \Pr[\text{success}]\ \Pr[\text{no arrival}]) \\ p_{kk} &= (1-\lambda)(1-p)(1-r) + p\,[(1-\beta)(1-r)+\beta r] \\ p_{kk+1} &= r(1-\lambda)(1-p) + \lambda(1-p)(1-r) + pr(1-\beta) \\ p_{kk+2} &= \lambda r(1-p) \end{aligned} \tag{1}$$

We can solve this Markov chain, by the usual techniques to find: i) the probability that the node under consideration, (say node i), is idle:

$$\pi_i\,(0) = 1 - \frac{\lambda_i + r_i}{p_i\,(\lambda_i + \beta_i)} \tag{2}$$

and ii) the average queue length at node i:

$$\overline{N}_i = \frac{\lambda_i}{\lambda_i + \beta_i} + \frac{r_i\,(1-r_i)\,\lambda_i\,(1-p_i)}{p_i\beta_i - \lambda_i\,(1-p_i) - r_i} \tag{3}$$

from which we can compute delays, etc., by Little's result.

Several parameters are still unknown, however.

a. The attempted transit arrival rate, λ_i. The relationship between successful and attempted transit arrivals is:

$$s_i = (1-b_i)\,\lambda_i + b_i\,(1-p_i)\,\lambda_i = (1-p_i\,b_i)\,\lambda_i$$

$$\Rightarrow \qquad \lambda_i = \frac{s_i}{1-p_i\,b_i} \tag{4}$$

b. The probability of success, β_i, can be computed by:

$$\beta_i = \sum_k \Pr[\text{destination is } k]\ \Pr[\text{no interference at } k] \tag{5}$$

Specific computation of β depends on the system assumptions and network topology. In the next section, we present the general approach for finding β and in the examples we present explicit expressions.

c. The 'busyness' probability, b_i. From the solution to the Markov chain, we have:

$$b_i = 1 - \pi_i\,(0) = \frac{\lambda_i + r_i}{p_i\,(\lambda_i + \beta_i)} \tag{6}$$

Which is a function of the other b_j's. This is therefore a set of (non-linear) simultaneous equations, which can be solved by iteration, see [7] for details.

Evaluation of the Interference

It is in this parameter, β, that the solution for Spread Spectrum differs from that for a non-Spread system. We must compute Pr[successsul transmission from node i], which depends on i) whether the destination transmits or not; ii) the number of other transmissions destined to the same destination; and iii) the total number of transmissions in the destination environment. The basic approach to determine the probability of k transmissions in the destination (j's) environment, can be expressed (using z-transforms) as:

$$M_{ij}(z) = \prod_{\substack{k \in N_j \\ k \neq i,j}} \left([1-b_k\,p_k] + b_k\,p_k\,z\right) \tag{7}$$

EXAMPLES

Paired Users - Infinite Threshold

The first example that we consider consists of a collection of n nodes that are paired-off, i.e., each node has a communication partner. We first consider an infinite receive threshold, so the pairs of users act independently. We can, therefore, compute the interference to be:

$$\beta_{ij} = 1-bp \tag{8}$$

i.e., there is inteference only if the destination node is busy and transmits. The

analytical and simulation results for a 10 node network, with transmission probabilities of $p=1/2$, are given in Table 1. We see relatively good agreement between the analytical predictions and simulation results, although the analysis under-estimates the delay in heavy load situations.

Inter-Arr Time	Network Load	Delay	
		Simulation	Analytical
4.0	0.25	2.19	2.11
2.0	0.50	2.26	2.24
1.0	1.00	2.84	2.62
0.8	1.25	3.17	2.90
0.5	2.00	6.13	4.95

Table 1: 10 Nodes, Infinite Receive Threshold ($p=0.5$)

A transmission probability of $p=0.5$ was used to avoid problems with instability, [14].

IAT	Threshold	Tx Prob	Simulation	Analytical
4.0	10	1/2	2.19	2.11
2.0	10	1/2	2.26	2.24
1.0	10	1/2	2.84	2.62
0.5	10	1/2	6.13	4.95
4.0	3	1/3	3.65	3.26
2.0	3	1/3	4.28	3.61
1.0	3	1/3	7.45	5.16
0.5	3	1/3	*	*
4.0	1	1/10	20.18	20.26
2.0	1	1/10	*	*

Table 2: 10 Nodes, Finite Thresholds

Paired Nodes - Finite Threshold

The next case we considered was the same traffic model, but with a reduced receive threshold. There is now some interaction between the different pairs. Since all nodes are identical, for a threshold of L, we have:

$$\beta = (1-bp) \sum_{k=0}^{L-1} \binom{n-2}{k} (bp)^k (1-bp)^{n-2-k} \qquad (9)$$

* Indicates that this system is overloaded

Table 2 shows the results of this case for several thresholds, we see the marked increase in traffic handling capability as the threshold increases. The transmission probabilities are again chosen in an attempt to force stability (although, the case of $L=3$ with $p=1/3$ will be unstable for certain traffic levels). It is interesting to determine the optimal transmission probabilities for this case (to maximise throughput).

Optimal Transmission Probabilities

For paired-off nodes and $L=10$, this is the same as a pair of nodes without Spread Spectrum, thus the optimum p (to maximise throughput) is given by:

$$\text{Max } p(1-p) \quad => p=1/2 \tag{10}$$

For a threshold of $L=1$, it is like a fully-connected non-Spread system, i.e.:

$$\text{Max } p(1-p)^9 \quad => p=1/10 \tag{11}$$

For other thresholds, it is more complex, e.g. for $L=3$:

$$\text{Max } p(1-p)[(1-p)^8 + 8p(1-p)^7 + 28p^2(1-p)^6] \quad => p \approx 0.219 \tag{12}$$

(thus for very heavy traffic ,we can expect $p=1/3$ to cause instability).

Centralised Network - No Capture

We now switch to a centralised traffic scenario. We consider a population of n (10) nodes with all traffic directed from nodes 1-9 to node 0 and a corresponding load from node 0 to nodes 1-9, thus the traffic load 0 to 9 and 9 to 0 is equal. For this case the two classes of traffic, inbound and outbound, must be treated separately. Using subscript 0 for the central node and using no subscripts for the other nodes, the interference for inbound traffic is:

$$\beta_{in} = (1-b_0p_0)(1-bp)^{n-2} \tag{13}$$

and for the outbound traffic:

$$\beta_{out} = 1-bp \tag{14}$$

If we incorporate the threshold effect, the inbound traffic is unaffected, but we must modify the expression for interference for the outbound traffic to be:

$$\beta_{out} = (1-bp) \sum_{k=0}^{L-1} \binom{n-2}{k} (bp)^k (1-bp)^{n-2-k} \tag{15}$$

It should be noted here that we are assuming that the central node is never able to capture one of a collection of simultaneous packets, they are all lost. We show the results for this case in Table 3. Selection of the transmission probabilties is more complex here and so we have included some cases that are unstable (bistable), the simulations results for these cases (where available) indicate that the system stayed in the 'good' state for the duration of the run. We again see fairly good agreement between the simulations and analytical models.

Inter Arr Time	L - Capture	Inbound			Outbound		
		p	Sim	An	p	Sim	An
8.0	10-N†	1/2	2.33	2.29	1/2	2.16	2.16
4.0	10-N†	1/2	3.24	2.72	1/2	2.43	2.39
2.0	10-N†	1/2	*	5.15	1/2	*	3.44
8.0	3-N	1/3	3.60	3.47	1/2	2.15	2.16
4.0	3-N†	1/3	4.68	4.16	1/2	2.34	2.39
2.0	3-N†	1/3	*	8.50	1/2	*	3.54
8.0	3-N	1/9	10.68	11.00	1/2	2.27	2.33
4.0	1-N	1/9	17.56	14.64	1/2	3.26	2.96
2.0	1-N	1/9	*	*	1/2	*	*

Table 3: 10 Nodes, Centralised Network, No Capture

Centralised Network - With Capture

Whereas in the previous case we assumed that if two transmissions were made simultaneously to the central node, both were lost, with capture we assume that one of them is captured. For this case, we have:

$$\beta_{in} = (1-b_0 p_0) \sum_{k=0}^{L-1} \binom{n-2}{k} (bp)^k (1-bp)^{n-2-k} \frac{1}{k+1} \qquad (16)$$

CONCLUSIONS

In this paper we have discussed extensions of our approximate model for the throughput/delay performance of broadcast networks to the spread apectrum environment. The results of this model are presented for several simple network configurations and comparison to simulation results are made. It is found that the agreement is reasonably good, although the results are not sufficiently accurate for accurate predictive purposes. The main use of this model would seem to be in determining approximate performance and identifying whether a particular set of parameters results in a stable network. The model is also useful for comparing different network topologies, access protocols and assumption about network operation.

REFERENCES

[1] "The Organization of Computer Resources into a Packet Radio Network", R.E. Kahn, IEEE Transactions on Communications, Vol COM-25, pp:169-178, January 1977.

[2] "The Spread Spectrum Concept", R.A. Scholtz, IEEE Transactions on Communications, vol. COM-25,pp:748-755, August 1977.

† - indicates that this system is bistable

[3] "Advances in Packet Radio Technology", R.Kahn, S.Gronemeyer, J.Burchfiel, and R.Kunzelman, Proceedings of the IEEE, Vol. 66, No. 11, pp:1468-1496, November 1978.

[4] "Performance of Slotted ALOHA Random Access with Delay Capture and Randomized Time of Arrival", D.H. Davis and S.A. Gronemeyer, IEEE Transactions on Communications, Vol. COM 28, No. 5, pp:703-710, May 1980.

[5] "Performance Analysis of Random Access Packet Switched Code Division Multiple Access Systems", Raychaudhuri, D., IEEE Transaction on Communications, Vol COM 29, No. 6, pp:895-901, June 1981.

[6] "An Iterative Scheme for Performance Modelling of Slotted ALOHA Packet Radio Networks", Lee, I., and J.A. Silvester, Proceedings of the ICC, pp:6G.1.1-5, 1982.

[7] "An Approximate Model for the Throughput Delay Performance of Multi-Hop Broadcast Networks: Slotted Aloha", Silvester, J.A., and I. Lee, To appear in the IEEE Transactions on Communications, also available as Technical Report Number CSI-83-09-01, Communications Sciences Institute, Department of Electrical Engineering, University of Southern California, 1983.

[8] "ALOHA Packet System with and without Slots and Capture", Roberts, L.G., ARPA Network Information Center, SRI, ASS Note 8, (NIC 112390), June 1972, also in Computer Communication Review, Volume 5, No. 2, April 1975, pp. 28-42, 1972.

[9] "A Spreading Code Protocol for a Distributed Spread Spectrum Packet Radio Network", E.S. Sousa and J.A. Silvester, to appear in proceedings of GLOBECOM 84, Atlanta, Georgia, December 1984.

[10] "Channel Access Protocols for Distributed Spread Spectrum Packet Radio Networks" E.S. Sousa and J.A. Silvester, submitted to IEEE Transactions on Communications (Sept 1984).

[11] "Interfering Queueing Processes in Packet Switched Broadcast Communication", L. Kleinrock and Y. Yemini, Proceedings of IFIP, 1980.

[12] "The Modelling of Packet Radio Networks", Segall, A., and A. Sidi, Technion Israel Institute of Technology, E.E. PUB 356, June 1979.

[13] "Analysis, Stability and Optimization of Slotted ALOHA with a Finite Number of Buffered Users", Saadawi, T.N., and A. Ephremides, IEEE Transactions on Automatic Control, Vol. AC-26, No. 3, pp:680-689, June 1981.

[14] "Bistable behavior of ALOHA-type systems", A. B. Carleial and M. E. Hellman, IEEE Trans. Commun., vol. COM-23, pp:401-410, April 1975.

Inter Arr Time	L - Capture	Inbound			Outbound		
		p	Sim	An	p	Sim	An
8.0	10-Y	1/2	2.20	2.22	1/2	2.04	2.16
4.0	10-Y†	1/2	2.58	2.50	1/2	2.36	2.39
2.0	10-Y†	1/2	3.62	3.49	1/2	3.49	3.30
8.0	3-Y	1/3	3.51	3.36	1/2	2.17	2.16
4.0	3-Y	1/3	3.87	3.82	1/2	2.41	2.39
2.0	3-Y†	1/3	5.48	5.53	1/2	3.52	3.33
8.0	1-Y	1/9	10.75	11.00	1/2	2.28	2.33
4.0	1-Y	1/9	16.18	14.64	1/2	3.24	2.96
2.0	1-Y	1/9	*	*	1/2	*	*

Table 4: 10 Nodes, Centralised Network, with Capture

IMPROVEMENTS IN THROUGHPUT OF A CDMA PACKET RADIO NETWORK DUE TO A CHANNEL LOAD SENSE ACCESS PROTOCOL

FOUAD A. TOBAGI
JAMES S. STOREY
Computer Systems Laboratory
Department of Electrical Engineering
Stanford University, Stanford, CA 94305

ABSTRACT

Features of a Code Division Multiple Access (CDMA) spread spectrum radio channel are incorporated into a packet radio network model. In the model, the channel is able to support up to a maximum of L simultaneous transmissions with a non-zero probability of success. Infinite transmitter population and finite transmitter population models are considered. The topology is a single-hop fully connected network with identical users. Single hop throughput and probability of successful first transmission are derived. The effects of L and of the number of users M are shown. Results are given for bit error rates and packet header synchronization probabilities which are functions of the number of simultaneous transmissions. A channel load sense access protocol is introduced in which radios are blocked from transmitting when the channel is heavily loaded. Performance using this protocol is shown for zero and for non-zero propagation delay.

1. INTRODUCTION

For many years, various forms of spread spectrum signalling have been used in secure communications systems. The performance of code division multiple access (CDMA) systems which use forward error correction (FEC) coding has been analyzed by various authors. Raychaudhuri [RAYC81] analyzed several network performance measures for a slotted single hop network. Musser and Daigle [MUSS82] derived the throughput for a single hop unslotted network with fixed length packets. Pursley [PURS83] studied asynchronous frequency hopping systems with fixed length packets, to find the throughput and the probability of a packet being correctly received.

All of the aforementioned analyses use a network access protocol which is analogous to ALOHA; namely, if a packet arrives at an idle radio it is immediately transmitted. The network performance of ALOHA based systems often can be improved by introducing carrier sensing. In this paper, we investigate the improvement that is gained by using a channel load sense access protocol in a CDMA network. In this protocol, radios are allowed to begin transmitting a packet only if the total number of transmissions sensed on the channel is less than a threshold K.

We analyze specific fully connected configurations of a multi-hop model developed by Boorstyn and Kershenbaum [BOOR80] and Brazio and Tobagi [BRAZ84]. For these configurations, the single-hop throughput and the probability of correct packet reception are derived. The networks modeled are asynchronous with variable length packets that are generated by Poisson processes. Section 2 presents the system assumptions. Section 3 describes the basic model, which assumes that the bit error probability is 0 or 1 depending on the number of interfering transmissions, and that each radio transmits to its dedicated receiver, which is always available. We present the probability of a packet being successful and throughput results for this model in Section 4. Section 5 presents a refinement of the model, allowing for a bit error rate (BER) which is a general function of the number of simultaneous transmissions. In Section 6, we relax the assumption that receivers are dedicated to transmitters. Section 7, investigates the performance of networks with channel load sensing, and the degradation due to propagation delay is analyzed in Section 8.

2. SYSTEM ASSUMPTIONS

The networks being modeled have the following features. The topology is fully connected with only single hop traffic. All users are identical. For M user finite population models, each user has a directed link to each of the $M-1$ other users, with equal traffic requirements on all $M(M-1)$ of these links.

The networks use spread spectrum CDMA radio channels. Receiver directed codes are used. The destination receiver has a probability α_t of synchronizing to the packet header (capturing the packet), where α_t is a function of the number of radios transmitting at the start of the packet transmission. Once capture occurs, the receiver never loses synchronization until the packet is completed.

If FEC coding is used, we assume that the effect of the coding can be incorporated into the channel BER performance. Consider a tagged packet with j transmissions on the channel. In the absence of

jamming and thermal noise, the symbol errors of the tagged packet will be due only to the $j-1$ other transmissions. For each bit, there is a probability $P_c(j)$ that this multi-user interference will not cause enough symbol errors to cause a bit error. In the scope of this paper, we limit our results to several simple cases for $P_c(j)$. Because we are considering the corrected bits, a single error will cause the packet to be unsuccessful. We assume an upper bound L, such that more than L simultaneous transmissions will cause all packets to be unsuccessful with probability 1.

We ignore the effect of acknowledgments, assuming a perfect and instantaneous acknowledgement channel is available.

A packet which is scheduled for transmission is blocked (i.e., rescheduled) if the radio is currently transmitting or receiving. In the channel load sense models, the transmission is also blocked if the number of radios currently transmitting is greater than or equal to a threshold K. Packets that are not blocked are immediately transmitted at the scheduling point.

For channel load sense models, each radio has perfect knowledge of the number of radios currently transmitting. At a scheduling point, the decision whether or not to transmit is based on this knowledge. Two possible implementations of channel load sensing are to estimate the number of active transmitters from the number of symbol errors corrected by the FEC decoding hardware or to estimate the number from the received pseudo-noise power, for example by using the front end AGC voltage. Clearly, a real implementation will perform worse than this idealized model.

3. BASIC MODEL

In the basic model, we consider receivers which are dedicated to transmitters, so that transmissions are never blocked due to the source being busy receiving, and transmissions are never unsuccessful due to the destination being busy. For finite transmitters, each user generates scheduling points from a Poisson process at the rate of offered traffic λ. We use the heavy traffic assumption that there is a packet available for transmission at every scheduling point. If the user is not already transmitting and the protocol does not inhibit transmission (in the channel load sense protocol), the transmission of the packet will begin at the scheduling point. For infinite transmitters, the scheduling points for all transmissions are generated from a Poisson process with aggregate rate λ. Again, a packet will begin to be transmitted at the scheduling point if the protocol does not inhibit transmission.

The transmission time of the packets is exponentially distributed with mean $1/\mu$. We use g to denote the normalized rate of offered traffic, $g = \lambda/\mu$. In this basic model, we assume a $P_c(j)$ which is either 1 or 0. Thus, channel behavior is such that no errors occur for L or fewer radios transmitting, and errors occur with probability 1 for more than L simultaneous transmissions.

Let $X(\tau)$ be the number of radios transmitting at time τ (We use τ to denote time throughout the paper to avoid confusion when we use t as the number of active transmitters in Sections 6, 7, and 8). $X(\tau)$ is a continuous time Markov chain. The state space S of $X(\tau)$ is $S = \{0, 1, 2, ...\}$ for infinite users, and $S = \{0, 1, 2, ..., M\}$ for finite transmitters. In both cases, S has a strictly positive stationary distribution $\{\pi_j, j \in S\}$. $X(\tau)$ for the infinite population model has the same state transition rates as the model of the $M/M/\infty$ queue, and for the finite transmitter case, the same as the $M/M/\infty//M$ queue. For infinite users,

$$\pi_j = \frac{(\lambda/\mu)^j}{j!} e^{(-\lambda/\mu)} \qquad j = 0, 1, 2, ... \tag{1}$$

For finite users,

$$\pi_j = \frac{(\lambda/\mu)^j}{(1+\lambda/\mu)^M} \binom{M}{j} \qquad j = 0, 1, ..., M \tag{2}$$

4. BASIC MODEL ANALYSIS

4.A. Throughput Analysis for the Basic Model

For a CDMA channel, it is possible for a packet to be successfully transmitted even if overlapping transmissions occur. For infinite transmitters, throughput S is defined as

$$S = \sum_{j=0}^{\infty} j \cdot (\text{Fraction of time spent transmitting } j \text{ successful packets}) \tag{3}$$

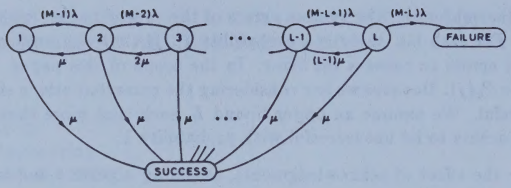


Fig. 1. The auxiliary Markov chain in the basic model.

For finite transmitters, S_i, the throughput for user i, is the fraction of time spent transmitting successful packets. Network throughput can be found as the sum of user throughputs, which is MS_i since all users are identical.

The probability of a packet being successful depends on its length as well as the state of the network upon its arrival. Furthermore, a packet cannot be counted as successful until it is completed, as an error anywhere will cause the entire packet to be unsuccessful. Thus, for determining success or failure, it is not sufficient to know only the state of the Markov chain at the beginning of a transmission. For this reason, we follow the approach used by Brazio and Tobagi [BRAZ84], and introduce an auxiliary Markov chain which traces the evolution of a tagged packet until either an error occurs or the packet transmission is successfully completed.

We define the following parameters:

T_s is the successful length of a packet, which is 0 if the packet is unsuccessful, and is the packet's length if it is successful.

$E(T_s|j)$ is the expected successful length of a packet given that the network is in state j upon its arrival, and that the packet was transmitted.

P_S is the probability that a packet will be successfully transmitted on the first attempt.

$P_{S|j}$ is P_S conditioned on the network being in state j upon arrival.

As derived in [STOR84], the probability of successful first transmission and throughput are

$$P_S = \begin{cases} \sum_{j=0}^{L-1} \pi_j P_{S|j} & \text{infinite transmitters} \\ \dfrac{\sum_{j=0}^{M-1}(M-j)\pi_j P_{S|j}}{\sum_{j=0}^{M-1}(M-j)\pi_j} & \text{finite transmitters} \end{cases} \tag{4}$$

$$S = \begin{cases} \sum_{j=0}^{L-1} \lambda \pi_j E(T_s|j) & \text{infinite transmitters} \\ \sum_{j=0}^{\min(L,M)-1} \lambda \pi_j (M-j) E(T_s|j) & \text{finite transmitters} \end{cases} \tag{5}$$

To account for header synchronization, we define α_j to be the probability that the receiver captures the packet given j interfering transmissions. The contribution to throughput by packets that found the network in state j upon arrival will be reduced by the factor α_j. For finite transmitters, we have

$$S = \sum_{j=0}^{\min(L,M)-1} \lambda \pi_j (M-j) \alpha_j E(T_s|j) \tag{6}$$

and similarly for infinite transmitters.

If $j \geq L$, $P_{S|j} = 0$ and $E(T_s|j) = 0$. To solve for $P_{S|j}$ and $E(T_s|j)$ for $j < L$, we construct an auxiliary Markov chain, as in [BRAZ84]. This chain consists of all the states that could be visited by a successful packet, and two absorbing states, Success and Failure. This chain is shown in Fig. 1 for the

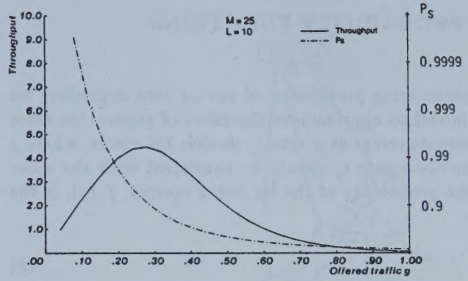


Fig. 2. Throughput and P_S versus Offered traffic g.

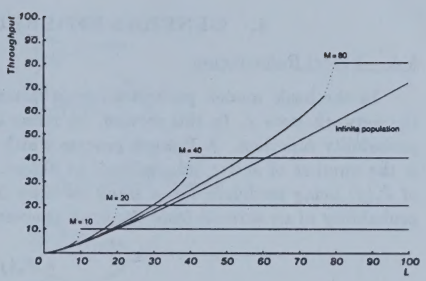


Fig. 3. Throughput versus L.

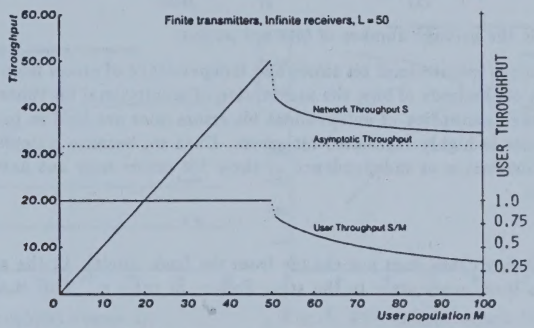


Fig. 4. Throughput versus User population M.

finite transmitter case. For the infinite transmitter model, the transitions from state j to state $j+1$ will be simply λ. We index the states Success and Failure as the last two states, so the transition rate matrix for this chain is

$$\mathbf{R}^* = \begin{pmatrix} \mathbf{R} & \mu\underline{\mathbf{1}} & \varphi \\ \underline{\mathbf{0}} & 0 & 0 \\ \underline{\mathbf{0}} & 0 & 0 \end{pmatrix} \tag{7}$$

where $\mathbf{R}$ is the $L \times L$ sub-matrix corresponding to transitions between the states $\{1, 2, ..., L\}$ of the auxiliary Markov chain. As shown in [STOR84] and [BRAZ84],

$$P_{S|j} = [-\mu\mathbf{R}^{-1}\underline{\mathbf{1}}]_{j+1} \qquad E(T_s|j) = [\mu\mathbf{R}^{-2}\underline{\mathbf{1}}]_{j+1} \qquad \text{for } 0 \le j < L$$

4.B. Results

In Fig. 2, we plot P_S and S for finite transmitters ($M = 25$) with $L = 10$. For $L < M$, the maximum throughput is limited to L. For values of $L \geq M$, no errors will occur, so for $g \to \infty$, all users will be constantly transmitting successfully, giving a throughput of M.

Maximizing S with respect to offered traffic g, gives the capacity $S_{\max}$, which is plotted in Fig. 3 for the infinite user case and for several values of M in the finite transmitter case. Although L is limited to whole numbers, a continuous function is shown in the following plots to make them easier to read. It can be seen that there is more than a proportional increase in throughput as L increases. Also, we again notice that for $L \geq M$ no errors ever occur, so there is a discontinuous jump in the throughput as L increases from $M-1$ to M.

For a given L, the optimum number of transmitters M is simply $M = L$. In Fig. 4, we plot the network throughput S and the user throughput (S/M) vs. M for the case $L = 50$. For $M = 50$ users, the network throughput is much higher than for larger M. In Section 6, we show that this is not true in the case of finite receivers.

5. GENERAL BIT ERROR PROBABILITY FUNCTIONS

5.A. Model Refinements

In the basic model, packets have an instantaneous error probability of one or zero depending on the network state j. In this section, we refine the model to approximate the effect of general bit error probability functions. A Poisson process which generates errors at a rate ϵ_j models bit errors, where j is the number of active transmitters as before. The error rate ϵ_j should be consistent with the value of $P_c(j)$ being modeled. For a small bit time Δt, the probability of the bit being correct, $P_c(j)$, is the probability of no arrivals from the error process, so

$$P_c(j) = e^{(-\Delta t\, \epsilon_j)} \tag{8}$$

$$\epsilon_j = -\frac{1}{\Delta t} \ln P_c(j) \quad \text{or} \quad \frac{\epsilon_j}{\mu} = -\frac{\ln P_c(j)}{\mu \Delta t} = -b \ln P_c(j), \tag{9}$$

where $b = 1/(\mu \Delta t)$ is the average number of bits per packet.

This model assumes infinitesimal bit times and independence of errors from bit to bit. For packets with average lengths of hundreds of bits, the assumption of infinitesimal bit times will be a good approximation. However, the assumption of independent bit errors does not hold in most systems, as forward error correction results in highly correlated bit errors. Even so, because a single bit error destroys the entire packet, the correlation or independence of these bit errors may not have much impact on the resulting throughput.

5.B. Analysis

The original Markov chain does not change from the basic model. In the auxiliary Markov chain, there are transitions from every state to the state Failure at rates ϵ_j. This changes only the diagonal elements of $\mathbf{R}$.

5.C. Results

Throughput was calculated for a simple family of $P_c(j)$ curves. The functions used are ramp functions, all of which start with $P_c(j) = 1$ for $j \leq L$, and decrease linearly to zero at $j = L^* + 1$. The number of bits per average Hacket was varied from 500 to 1000. It was found that there was little variation in the throughput as L^* was varied. This makes sense intuitively, since even one error will cause the packet to be unsuccessful, and when in a state j with $P_c(j)$ less than one, there will be a high probability of at least one error after many bits. Thus, the difference between a state with $P_c = 0$ and one with $P_c \neq 0$ but $P_c < 1$ will be small, except for P_c very close to 1.

Although the step function is only a first order approximation of the actual behavior of CDMA radio channels, the ramp function used in this section is not a more realistic model. For this reason, throughout the rest of this paper, we set aside this extension and consider only step function $P_c(j)$ curves, i.e., $P_c(j) = 1$ for $j \leq L$ and $P_c(j) = 0$ for $j > L$.

6. FINITE RECEIVERS

6.A. Analysis

In this section, a finite user network is modeled, in which both the sources and the destinations of packets are among the M users. Several modifications are needed. First, if the source is busy receiving it will not transmit a packet which is scheduled. Second, if the destination is busy receiving or transmitting, the packet will not be successful. Finally, idle receivers which do not capture a packet affect the state transitions.

The state variable $\mathcal{X}(\tau)$ is now defined as both the number of active transmitters t, and the number of receivers r which have captured a packet. The state space is $S = \{(t, r) : 0 \leq t \leq M, 0 \leq r \leq t, \text{ and } t + r \leq M\}$. The form of the auxiliary Markov chain is given in Fig. 5. The steady state probabilities $\pi_{(t,r)}$ of the Markov chain $\mathcal{X}(\tau)$ are found numerically from the conservation equations

$$\mathbf{Q}\underline{\pi} = 0 \qquad \text{and} \qquad \sum_{(t,r) \in S} \pi_{(t,r)} = 1,$$

where $\mathbf{Q}$ is the transition rate matrix.

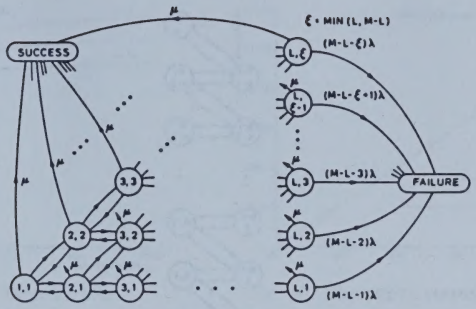


Fig. 5. The auxiliary Markov chain for the finite receivers model.

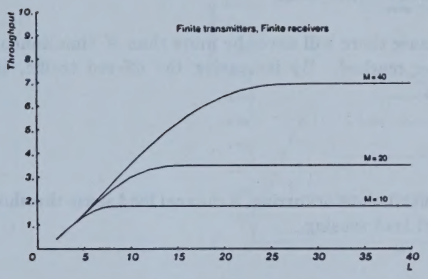


Fig. 6. Throughput versus L.

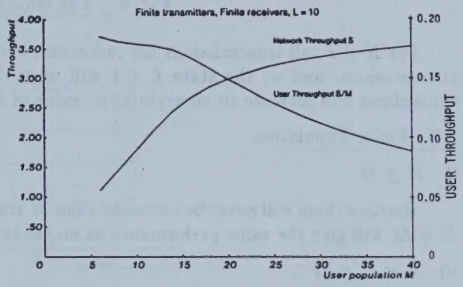


Fig. 7. Throughput versus User population M.

We define the variables α_t and $E(T_s|(t,r))$ similarly to the variables in the basic model.

As shown in [STOR84],

$$S = \sum_{(t,r)\in S'} \lambda \pi_{(t,r)}(M-t-r)\left(1-\frac{t+r}{M-1}\right)\alpha_t E(T_s|(t,r)) \tag{10}$$

$E(T_s|(t,r))$ is again found from the auxiliary Markov chain transition rates.

6.B RESULTS

$S_{\max}$ vs. L is plotted for several values of M in Fig. 6. The throughput is much lower than for the same values of M in the basic model. This is expected, as the maximum possible throughput is $\lfloor M/2 \rfloor$. In Fig. 7, we plot $S_{\max}$ vs. M for $L = 10$. The user throughput is fairly constant for $M \lesssim 2L$, and drops off for larger M. However, contrary to the infinite receiver case, the network throughput increases monotonically with M even for $M > 2L$.

7. CHANNEL LOAD SENSING

Many authors have shown that for ALOHA-like systems, the network performance can be improved by sensing the channel and blocking transmissions when the channel is busy. In a CDMA network where multiple users can transmit simultaneously, similar improvement is possible if the radios can sense the number of radios transmitting and block transmission when the channel is heavily loaded.

7.A Infinite User Population

We use K to denote the number of sensed transmissions at which new transmissions are blocked. For $K > L$, the state space S is just a truncation of the basic model state space. This has the same state transition rates as the state space of the $M/M/K/K$ queue, which is a K server loss system [KLEI75]. The auxiliary Markov chain is the same as in the basic model, so for this range of $K > L$, the only difference from the basic model is in the initial state probabilities $\{\pi_j\}$,

$$\pi_j = \frac{(\lambda/\mu)^j/j!}{\sum_{k=0}^{K}(\lambda/\mu)^k/k!} \tag{11}$$

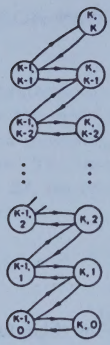


Fig. 8. The truncated Markov chain.

For $K \leq L$, all transmissions are successful, because there will never be more than K simultaneous transmissions, and so the state $L+1$ will never be reached. By increasing the offered traffic, the throughput will increase to an asymptotic value of K.

7.B Finite Population

i) $K \geq M$

Because there will never be any more than M transmissions occurring, a channel load sense threshold $K \geq M$ will give the same performance as no channel load sensing.

ii) $L < K < M$

In this case, the state space is truncated at $t = K$, while the auxiliary Markov chain remains the same. Consequently, the only difference from the case studied in Section 6 is that the initial probability distribution $\{\pi_{(t,r)}\}$ changes. Thus, the solution can be found similarly to Section 6.

iii) $K \leq L$ and $K < M$

For this range of K, all captured packets will be successful. However, packets may not be captured, due to the destination radio being busy. Even so, we note that when K is small compared to M, the probability of no capture is very small, so the throughput is maximized by a very large g. Unfortunately, the numerical solution used for finding throughput becomes unstable as g becomes large. Therefore, we use an approximation of the Markov chain which is valid for such values of g. For very large g, we expect the number of transmissions to be K, since as soon as a transmission ends a new scheduling point will be generated and transmission of a new packet will begin. However, in order to find the throughput, we must find the distribution of the number of captured receivers.

This distribution is found by considering a truncated Markov chain consisting of states $\{(t,r) : t = K-1 \text{ or } t = K\}$, as in Fig. 8. Because all captured packets are successful, we can find the throughput directly as

$$S = \sum_{r=0}^{K} r\pi_{(K,r)} \tag{12}$$

For large K, the maximum S is achieved at small values of g for which this approximation is no longer valid. However, for g in this range, we can use the numerical solution developed in Section 6.

7.C Results

For infinite users and $L = 20$, we plot S vs. g for various values of K, and also for no channel load sensing in Fig. 9. For $K \leq L$, the throughput increases towards K as g increases. Also, the maximum throughput for $K = L+1$ is almost equal to the maximum for no channel load sensing. We then plot the maximum throughput $S_{\max}$ vs. the channel load sense point K for infinite users in Fig. 10, varying L from 20 to 50. Fig. 10 clearly shows how little improvement there is for $K = L+1$. Thus, we find that performance is very sensitive to the channel load sense point, and that it is better to operate below the optimum K than above it.

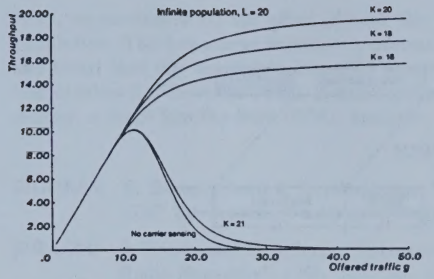


Fig. 9. Throughput versus Offered traffic. for several values of K.

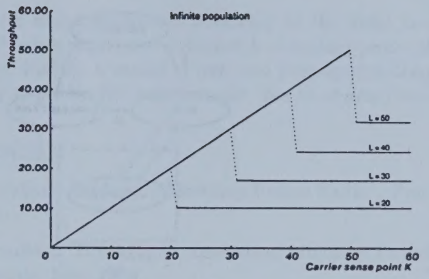


Fig. 10. Througput versus Carrier sense point

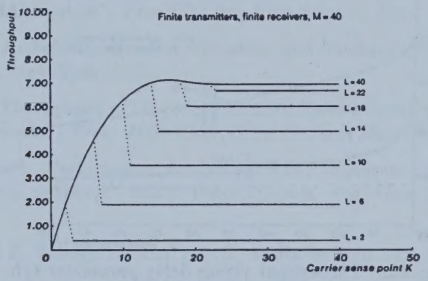


Fig. 11. Throughput versus Carrier sense point.

For finite transmitters and receivers, we plot in Fig. 11 the throughput vs. the channel load sense point K for $M = 40$, using various values of L. For $K \leq L$ and $K \leq 17$, S was calculated from Eqn. 12. For small values of L, performance is again sensitive to the channel load sense point, with significant improvement for $K = L$, and almost no improvement for $K = L + 1$. As L increases, the optimum K no longer equals L, but remains at somewhat less than $M/2$. For L approaching M, there is little improvement due to channel load sensing.

8. PROPAGATION DELAY

It is well known that carrier sensing systems degrade as propagation delay becomes large relative to the packet lengths. To investigate this effect for the CDMA model, we introduce an exponentially distributed holding time at the beginning of each packet. The decision on whether or not to transmit is again based on the number of radios which can be sensed transmitting, but this decision does not account for users in the holding state. This model approximates the behavior due to propagation delay.

From the standpoint of a receiver, all other users have a state model which is diagrammed in Fig. 12(a). During the holding state, the receiver cannot detect the transmission. From the standpoint of a transmitter, the holding time and the transmission time are reversed. Thus, a packet transmission can be illustrated as in Fig. 12(b). The model has the peculiarity that a radio is busy both during the transmission time and during the holding time, so it is as if a radio continues to be busy from the time a packet transmission is completed until the time the packet finishes propagating to the receiver. This is shown as the shaded region in Fig. 12(b). This extra busy period causes the throughput to be reduced even for no channel load sensing.

8.A. Delay Model

We introduce the parameters ν, where $1/\nu$ is the average holding time. and $h = \nu/\mu$, so $1/h$ is the average holding time normalized by the mean packet length. This parameter $1/h$ is equivalent to the standard delay measure a, the propagation delay normalized by the packet length.

The number of transmitters in a holding state is w, so the Markov chains are as follows. For infinite receivers, the states are indexed by w and t. The state space is $S = \{(w,t) : 0 \leq w \leq M, 0 \leq t \leq M - w\}$.

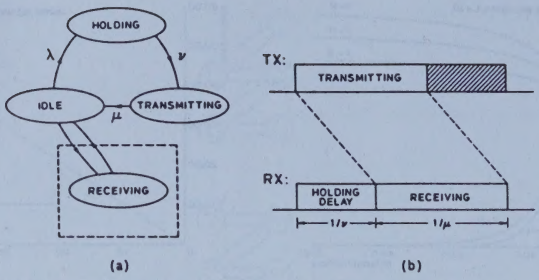


Fig. 12. User's state model incorporating propagation delay.

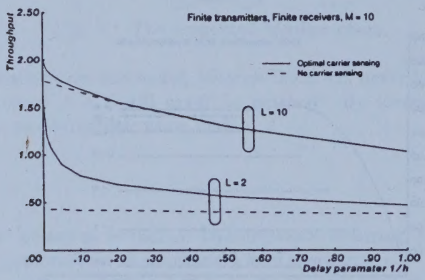


Fig. 13. Throughput versus delay parameter $1/h$.

For finite receivers, the states are indexed by w, t, and r;

$$S = \left\{ \begin{matrix} 0 \le w \le M \\ (w,t,r) : 0 \le t \le M - w \\ 0 \le r \le M - w - t, \text{ and } r \le t \, . \end{matrix} \right\}$$

Channel load sensing causes the deletion of transitions due to new packet transmissions for $t \ge K$.

As derived in [STOR84], for infinite receivers,

$$S = \sum_{(w,t) \in S'} \nu \pi_{(w,t)} w \alpha_t E\big(T_s | (w,t)\big) \tag{13}$$

Again, $E\big(T_s|(w,t)\big)$ is found from $\mu \mathbf{R}^{-2}\underline{1}$ for $t < L$. For finite receivers,

$$S = \sum_{(w,t,r) \in S'} \nu \pi_{(w,t,r)} w \left(1 - \frac{w+t+r-1}{M-1}\right) \alpha_t E\big(T_s | (w,t,r)\big) \tag{14}$$

8.B. Results

Fig. 13 is a plot of the maximum throughput vs. $1/h$ for several values of L. These results were calculated for no channel load sensing and for channel load sensing with K at the optimum channel load sense point. For no channel load sensing, the throughput decreases as $1/h$ increases, due to the users being busy during the holding periods. Also, as expected, the significant improvement from channel load sensing for L small relative to M degrades as $1/h$ increases. The improvement drops quickly for small delay ($1/h < 0.10$). Nevertheless, even for large delays ($1/h \approx 1.0$), for L small relative to M, the throughput is increased with channel load sensing.

9. CONCLUSIONS

In this paper, we developed the Markovian model of a packet radio network for several specific fully connected topologies. For a basic model, we found the throughput and the probability of successful first transmission, varying the multiple user capacity L and the number of transmitters M. We introduced extensions which allow general bit error rate functions and packet header synchronization probabilities.

Next, we accounted for the effect of both the receivers and transmitters belonging to the finite user population. This had a large impact on performance. We then considered a channel load sensing protocol, and found that this resulted in improved performance. Finally, a model of non-zero propagation delay was developed. We showed that increasing the delay degrades the improvement due to channel load sensing, a result familiar from CSMA analyses.

REFERENCES

[BOOR80] R. Boorstyn and A. Kershenbaum, "Throughput Analysis of Multihop Packet Radio", *Proc. ICC '80*, Seattle, Washington, June 1980.

[BRAZ84] J. Brazio and F. Tobagi, "Theoretical Results in Throughput Analysis of Multihop Packet Radio Networks", *Proc. ICC '84*, Amsterdam, May 1984.

[KLEI75] L. Kleinrock, *Queueing Systems*, Vol. 1, Wiley, New York, 1975.

[MUSS82] J. Musser and J. Daigle, "Throughput Analysis of an Asynchronous Code Division Multiple Access (CDMA) System", *Proc ICC '82*, Philadelphia, June, 1982.

[PAPO65] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 1965.

[PURS83] M. Pursley, "Throughput of Frequency Hopped Spread-Spectrum Communications for Packet Radio Networks", *Proc. 1983 CISS*, John Hopkins Univ., March 1983.

[RAYC81] D. Raychadhuri, "Performance Analysis of Random Access Packet-Switched Code Division Multiple Access Schemes", *IEEE Trans. Comm.* Vol. Com-29, No. 6, pp. 895-900, June 1981.

[STOR84] J. Storey and F. Tobagi, Internal report, available from authors

ACKNOWLEDGEMENT

This work was supported in part by the Defense Advanced Research Projects Agency under Contract MDA 903-79-C-0201, Order A03717, monitored by the Office of Naval Research. James S. Storey is on a fellowship from the Hughes Aircraft Company.

A CHANNEL ACCESS PROTOCOL FOR SURVIVABLE RADIO NETWORKS WITH FREQUENCY HOPPING SPREAD SPECTRUM SIGNALING

D. BAKER and J. WIESELTHIER
Naval Research Laboratory
Washington, D.C. 20375

and

A. EPHREMIDES
Electrical Engineering Dept.
University of Maryland
College Park, MD. 20742

and

Naval Research Laboratory

ABSTRACT

We describe an adaptive, time-division-multiple-access/code-division-multiple-access (TDMA/CDMA) protocol for controlling channel access in a survivable radio network with frequency hopping (FH) spread spectrum signaling. In a frequency hopping spread spectrum network, CDMA techniques permit successful simultaneous transmission over a wideband channel by a number of users. Nevertheless, due to propagation delays, FH signals that might otherwise be orthogonal can interfere with each other (secondary collisions). Provided the number of interfering signals is not too great, error correcting coding can be used to achieve acceptable performance levels. Our protocol is designed to achieve high channel throughput while at the same time keeping the level of secondary collisions low. We describe a fully distributed algorithm that dynamically creates the transmission schedules that are the basis of this protocol. These schedules can adapt to varying traffic loads and changing network topologies. The intended application is a broadcast network, although point-to-point traffic can also be handled. We illustrate the operation of the distributed algorithm with a simulation example.

INTRODUCTION

The feasibility of constructing packet switched communication networks for mobile users has been demonstrated by the experimental Packet Radio Network [1,2]. What is not yet fully understood, however, is how to make Packet Radio and similar networks survivable. Survivability is, of course, a primary requisite for military communication networks [3]. The design of a network in which survivability is an important consideration must emphasize operation under worst case conditions -- when nodes and links may be failing and when traffic may be near its peak. It is common for such networks to employ distributed control -- to avoid the vulnerability to the loss of a key node -- and spread spectrum signaling -- to provide each link with some protection against jamming.

Spread spectrum signaling, introduced because of its antijam capability, leads naturally to the use of Code Division Multiple Access (CDMA) techniques. The type of CDMA that we consider in this paper is frequency hopping (FH). In FH-CDMA systems the code corresponds to the FH pattern. The purpose of this paper is to present a channel access protocol that is well suited for use in FH spread spectrum networks that must be survivable.

To activate a link in a FH spread spectrum network, it is necessary for the transmitting node to have access to the communication channel and

for both the transmitter and receiver to use the same FH pattern. Several algorithms to effect this coordination have appeared in the literature recently that are based on an adaptive time-division-multiple-access (TDMA) approach to channel access. In [4], a procedure is described whereby each node can develop its own TDMA schedules. These schedules govern when the node can communicate with each of its neighbors, and rules are given for selecting FH-codes for each link activation. However, these schedules are developed under the assumption of uniformly distributed traffic loads among the nodes. In addition, no attempt is made to limit the number of simultaneous transmissions in a given area. Hajek [5] showed how suitable schedules could be developed that had neither of the above-mentioned shortcomings. However, it is not known how to implement the procedures given in [5] in a network in which communication may be unreliable as a result of jamming or node loss.

Another approach to traffic-adaptive TDMA scheduling is the Handoff Assigned Multiple Access (HAMA) protocol [6]. Under HAMA, nodes may dynamically hand-off their own TDMA slot to a neighboring node. The protocol can handle varying traffic loads, but for sparsely connected networks the channel throughput is not much better than that which would be achieved using fixed-TDMA.

In this paper, we describe a channel access protocol that is suited to high traffic environments, is traffic adaptive, interference-limiting, and performs well in both highly- and sparsely-connected networks. This protocol is intended for use in networks whose traffic is primarily broadcast, although it can be used in networks with point-to-point traffic. Our protocol is based on a specific network architecture, called the Linked Cluster Architecture, which is briefly described in the next section. Following that we present the basic concepts of our protocol and give details of a fully distributed algorithm that can be used to implement our approach. This is followed by a simulation example that is used to illustrate various features of the protocol. Next, prior to our conclusions, we outline an extension of the basic protocol that would allow new nodes to enter the network.

THE LINKED CLUSTER ARCHITECTURE

Clustering has been suggested as a technique for organizing some networks. For example, Kleinrock and Kamoun [7] show that by organizing a large network into a hierarchy of node-clusters it is possible to realize substantial savings in the length of routing tables. Baker et al [4] have pointed out several advantages of a one-hop cluster structure in which all cluster members are directly connected to a clusterhead node. For example, clusterheads can serve as regional broadcast nodes and as local coordinators to enhance channel throughput. Our approach is based on this one-hop cluster architecture, which is also referred to as a Linked Cluster Architecture.

An example of a network organized into linked clusters, using the approach of [4], is shown in Figure 1. Nodes can assume one of three roles -- clusterhead, gateway, or ordinary node. Each node-cluster has a clusterhead, which is bidirectionally linked to every other node in the cluster. Gateways are specially designated relays for passing traffic between clusterheads. The clusterheads, gateways, and the links between them form a backbone network. The backbone network may be used to send network control traffic as well as user traffic. In addition, the network can use any other links to pass traffic -- we have not shown all the possible links

in our example. In this structure, every node is either bidirectionally linked to a clusterhead or is itself a clusterhead. Also, clusters tend to be spread out because clusterheads are never directly connected. Neighboring clusters are either overlapping -- i.e., some nodes are members of both clusters -- or non-overlapping -- i.e., no nodes belong to both clusters but at least one node from one of the clusters is directly linked to at least one node of the other cluster.

A method has been developed for implementing the linked cluster structure in a network [4]. This method uses a distributed algorithm, called the Linked Cluster Algorithm (LCA), to control the selection of clusterhead and gateway nodes. This algorithm contains a communication part and a computation part. Communication is governed by a fixed-TDMA schedule. To adapt to changing connectivities, the algorithm is run periodically -- new clusters taking the place of old ones as connectivities change. Timing for network reorganization is shown in Figure 2 for a network of N nodes. Two TDMA frames are needed to complete the algorithm. During the first frame, connectivities are tested. In the second frame, each node announces its bidirectional connectivities along with the identity of its own clusterhead. Each node decides, just prior to its broadcast in Frame 2, whether or not it should become a clusterhead. The rule is that a node becomes a clusterhead if it is not already bidirectionally linked to one.

THE TRANSMISSION SCHEDULING ALGORITHM (TSA)

The channel access protocol that we present in this paper adds additional transmission schedules to the fixed-TDMA schedule used in the implementation of the Linked Cluster structure and shown in Figure 2. In a frequency hopping spread spectrum network, CDMA techniques permit successful simultaneous transmission over a wideband channel by a number of users. Nevertheless, due to propagation delays, FH signals that might otherwise be orthogonal can interfere with each other (secondary collisions). Provided the number of interfering signals is not too great, error correcting coding can be used to achieve acceptable performance levels [8]. Our protocol is based on the use of transmission schedules that are designed to achieve high channel throughput while at the same time keeping the level of secondary collisions low. To avoid the vulnerabilities associated with a central controller, we propose to form these schedules by a distributed algorithm, called the Transmission Scheduling Algorithm (TSA), which we describe in this section.

Several performance measures can be applied to evaluate such scheduling algorithms and the resultant schedules including:

- effects of missed communications (both during and after running the algorithm),
- effects of topology changes,
- receiver utilization efficiency,
- number of schedule conflicts,
- number of primary collisions (i.e., signals received on the same FH-code) and secondary collisions,
- adaptivity to varying traffic loads,
- time to respond to changes,
- ability to handle both point-to-point and broadcast traffic,
- fairness,
- overhead,
- throughput, and
- delay.

Our protocol addresses all of these considerations, as will become apparent later on.

Our protocol is an extension of the fixed-TDMA approach to channel access. For the applications that we are considering, the most important feature of fixed-TDMA is that it offers a means of coordinating transmission and reception that is immune to disruption by node loss, link loss, or high traffic levels. On the other hand, fixed-TDMA has a well-known list of shortcomings including:

1) difficult to add nodes,
2) not adaptive to varying traffic,
3) inefficient for sparsely connected networks,
4) frame size is large for networks of many nodes, and
5) poor response under light loading.

Our channel access protocol overcomes shortcomings 1-3 above and, in sparsely connected networks, lessens the impact of items 4 and 5.

We will use the three-cluster network shown in Figure 3 to illustrate our method for developing transmission schedules. The basic idea is to establish separate transmission schedules for each cluster in addition to the fixed-TDMA schedule that we assume is always in effect for the network as a whole. The nodes in the network are numbered from 1 to N. Whenever a node's number appears in a slot of a schedule, that node can transmit during that slot. Thus, if there are M clusters, there are at most M+1 simultaneously transmitting nodes, and each cluster must always contain at least one transmitting node. Since there are fewer clusters than there are nodes in the network and since clusters tend to be spread out, the number of secondary conflicts will usually be kept small.

Several choices exist concerning the selection of a FH code for use during transmission -- e.g., transmitter-based, receiver-based, or common (shared) code. The use of transmitter-based codes is appropriate for broadcast networks, while the use of receiver-based codes has some advantages for point-to-point communication. The use of a common code can avoid the problem of transmitter/receiver code coordination, but at the expense of requiring greater coordination to avoid conflicts between user transmissions. We have chosen to use transmitter-based codes for the following reasons:

- allows simultaneous transmissions by several nodes,
- avoids any transmission schedule conflicts (a node always transmits on its own transmitter code regardless of which schedule(s) it is broadcasting on), and
- can handle the case where transmitting node does not know who can hear it -- this can occur if a node is jammed.

The Transmission Scheduling Algorithm, which is used to develop the transmission schedules for each cluster, is a distributed algorithm that is piggybacked onto the Linked Cluster Algorithm in the sense that it shares the same Frame 2 transmission slot (see Figure 2). The information that is exchanged during the execution of the LCA and the additional information that is passed by the TSA during Frame 2 of the reorganization epoch are shown in Figure 4. Here, we focus on the TSA communication.

As part of the TSA, each clusterhead announces during Frame 2, the number of slots to be allocated to each node in its cluster. This set of allocations is used to obtain a transmission schedule for nodes in that cluster. The TSA does not explicitly state what criteria are to be used to determine the number of slots to be given to each node since the basis on which slots are allocated is expected to differ from one network to the next. In addition, the TSA does not specify the procedure to be followed to

develop the transmission schedule from the set of slot allocations. This too may differ among networks. However, in our simulation example we give an example of rules that might be used to perform these two tasks.

AN EXAMPLE

We illustrate the operation of the Transmission Scheduling Algorithm using the network whose connectivities are shown Figure 5a (all links are bidirectional). Applying the clustering techniques of [4], we obtain the three-cluster network shown in Figure 5b. Consequently, four transmission schedules are formed -- one fixed-TDMA schedule, which includes all of the network nodes, and three cluster schedules, each containing only the nodes that belong to the cluster.

The schedules are formed as follows. Node 1 announces, as part of its Frame 2 broadcast, the number of slots of its cluster schedule to be allocated to various cluster members. For the sake of simplifying the example, let's assume that nodes know the degree of connectivity of their neighbors and allocate slots based on this parameter. Thus, for example, node 1 forms the slot allocations 4,0,5,0,0,3,0,6,5,0 for nodes 1 through 10, respectively, and announces this allocation list -- i.e., node 1 would be assigned 4 slots, node 3 would get 5 slots, etc. Each node that receives node 1's broadcast performs the same operations upon this list to arrive at the same cluster schedule, which is shown in Figure 5c. The procedure used to generate the schedules from the allocation list in this example was as follows. First, the algorithm constructs an ordered schedule containing the appropriate number of slots allocated to each node. This schedule is then randomized -- each node applies the same randomization seed. The resultant schedules have their slot allocations scrambled as shown in Figure 5c. Clusterheads 2 and 4 follow procedures similar to that of clusterhead 1, and their schedules are also shown in Figure 5c. When it is node 3's turn to transmit in Frame 2, node 3 will rebroadcast the slot allocation list announced by node 1. In this way, node 7, which is a neighbor of 3 that happens not to be within range of node 1, can learn about the cluster-schedule transmissions of 3. Nodes 5,6,7, and 10 proceed in a manner similar to that used by node 3. Nodes 8 and 9 are somewhat special in that they are each members of more than one cluster. The TSA requires only that these nodes rebroadcast the slot allocation list of their own clusterhead, which is defined as the lowest numbered clusterhead to which they are bidirectionally connected. Thus, both 8 and 9 rebroadcast node 1's slot allocation list.

The period of each schedule may be different. In the example of Figure 5c, the schedules have periods of 10, 23, 14, and 10. Schedules formed during a reorganization epoch take effect at the end of that epoch. The schedules repeat until the end of the next network reorganization epoch, when new schedules take effect.

Two performance measures were computed for this example, namely, receiver utilization and the numbers of secondary collisions. Here, receiver utilization refers to the percentage of time that a receiver is within range of another transmitting node. For a fixed-TDMA protocol this number can be quite low -- e.g., 36% for the network shown in Figure 5a. On the other hand, the schedules produced by the TSA result in a much higher receiver utilization, namely 84% (measured over period of 80 slots). In our simulation model, we count as a secondary collision at a receiver a transmission from a node that is within communication range of the receiver but which is on a different FH-code from that which the receiver is monitoring. For the network in our example, the greatest number of secondary

collisions that can occur is 5, which can occur at node 8 if all of its neighbors are transmitting simultaneously. The schedules produced by the TSA reduce the maximum level of secondary collisions to 3. Actually, 98% of the time the number of secondary collisions were observed to be less than or equal to 2, and 81% of the time they were less than or equal to 1. Thirty-five percent of the time there were no secondary collisions.

ADDING NEW NODES

It is usually difficult to add new nodes to a network that uses a TDMA protocol. Changes to the TDMA frame should be done in such a way that:

- all the nodes affected are informed of the planned change
- all the nodes affected know when the change is to go into effect,
- the time to effect the change is short, and
- there is no interruption in service.

If the network is not fully-connected and if links are unreliable, it may be very difficult to satisfy all of these requirements. On the other hand, the way cluster schedules are formed by the TSA makes it easy to incorporate new nodes into them.

We suggest the following modification of the TSA to allow for the addition of new nodes into the cluster schedules. The fixed-TDMA frame is modified to include one or more contention slots. These are contention slots in the sense that a node transmitting during that slot uses a common (shared) FH-code, which is known to all nodes in the network. A node that wishes to enter the network must first monitor the channel to determine which nodes it can hear. At the first opportunity, the new node may transmit a "network-entry" packet during the contention slot. This packet contains (as a minimum) the new node's own identification number plus a list of the nodes that it can hear. Nodes that receive a network entry-packet, and which are also bidirectionally connected to the new node, save this packet until the next reorganization epoch occurs. At the next reorganization epoch, any clusterhead that has saved a network-entry packet may allocate slots in the new cluster schedule to the new node. A slightly revised slot-allocation list is then broadcast by the clusterhead. Figure 6 graphically depicts the proposed modifications.

CONCLUSIONS

We have focussed on the problem of channel access in survivable radio networks that use frequency hopping spread spectrum signaling. Our approach is based on a time-slotted, code-division-multiple-access system that uses a set of transmission schedules to control channel access. We show that a network that is organized into node clusters in which each cluster has a designated clusterhead that is bidirectionally linked to every cluster member provides a convenient framework on which to build a multi-schedule system. A fully-distributed Transmission Scheduling Algorithm (TSA) has been described that implements a multi-schedule system within such a cluster architecture. Transmitter-based frequency hopping (FH) codes are used to avoid scheduling conflicts.

Our channel access scheme uses both fixed-TDMA schedules and dynamic, cluster schedules. The fixed-TDMA schedules provide a non-adaptive but robust frame structure that is known and used by all nodes in the network,

while the cluster schedules provide adaptive frame structures but ones which are likely to be known and used only in the vicinity of each cluster. We have shown that new nodes can be easily incorporated into the cluster schedules.

REFERENCES

1. R.E. Kahn, S.A. Gronmeyer, J. Burchfiel, and R.C. Kunzelman, "Advances in Packet Radio Technology," Proc. IEEE, Vol. 66, Nov. 1978, pp. 1468-1496.

2. W. MacGregor, J. Westcott, and M. Beeler, "Multiple Control Stations in Packet Radio Networks," MILCOM '82, pp. 10.3-1 - 10.3-5.

3. B. Leiner, T. Klein, and B. Graff, "Data Distribution in a Tactical Environment," SIGNAL, Nov. 1983, pp. 27-29.

4. D.J. Baker, A. Ephremides, and J.A. Flynn, "The Design and Simulation of a Mobile Radio Network with Distributed Control," IEEE J. on Selected Areas Commun., SAC-2, No. 1, January 1984, pp. 226-237.

5. B. Hajek, "Balanced Scheduling in a Packet Synchronized Spread Spectrum Network," IEEE INFOCOM '83, pp. 56-65.

6. C. Warner, et al., "Handoff Assigned Multiple Access (HAMA): A New Multiple Access Protocol for Computer Communication Networks," MILCOM'83, pp. 183-189.

7. L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks," Computer Networks (1), 1977, pp. 155-174.

8. M. B. Pursley, "Throughput of Frequency-Hopped Spread Spectrum Communications for Packet Radio Networks," Proc. 1983 Conf. on Information Sciences and Systems, Mar. 23-25, 1983.

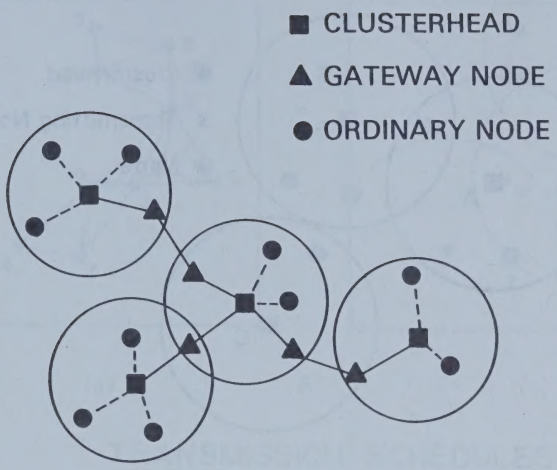

CLUSTERHEAD
GATEWAY NODE
ORDINARY NODE

Fig. 1

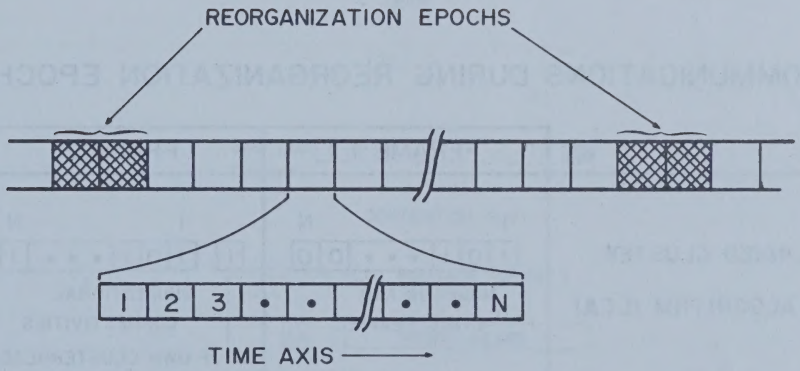

REORGANIZATION EPOCHS
1
2
3
N
TIME AXIS

Fig. 2

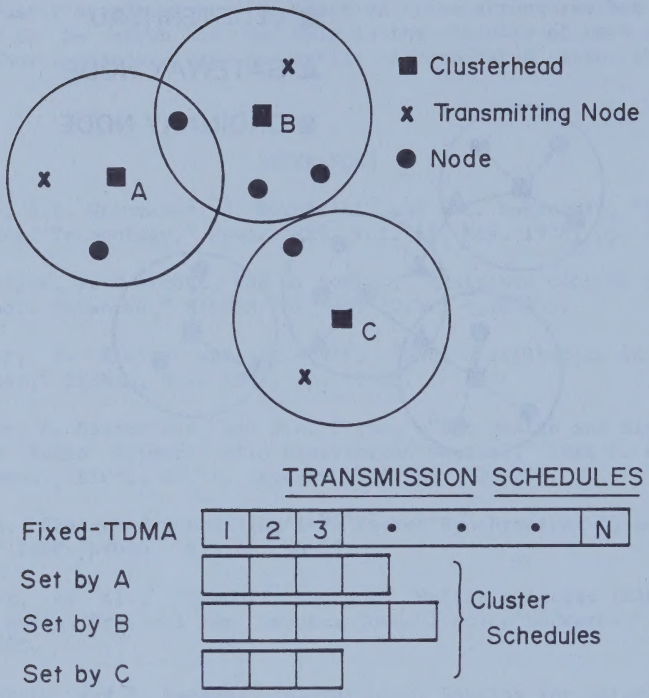
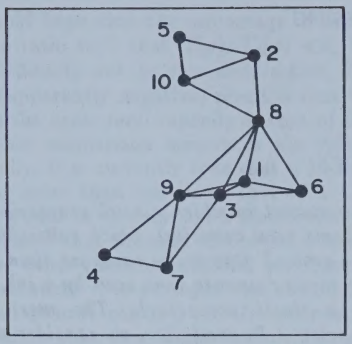


Fig. 3

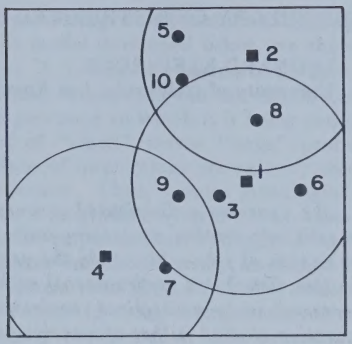
COMMUNICATIONS DURING REORGANIZATION EPOCH

	FRAME 1	FRAME 2
LINKED CLUSTER ALGORITHM (LCA)	I … N 1 0 1 • • • 0 0 NODES HEARD (THIS FRAME)	I … N 1 1 0 1 • • • 1 BIDIRECTIONAL CONNECTIVITIES ID. OF OWN CLUSTERHEAD
TRANSMISSION SCHEDULING ALGORITHM (TSA)	NONE	I … N 3 0 4 • • • 4 SLOT ALLOCATIONS

Fig. 4



(a)



(b)

TRANSMISSION SCHEDULES

	Slot #																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Fixed	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3
Set by 1	8	8	3	9	1	6	3	1	9	3	6	6	9	8	8	1	1	3	3	8	9	9	8
Set by 2	10	8	8	8	8	5	10	2	2	8	10	8	5	2	10	8	8	8	8	5	10	2	2
Set by 4	7	9	4	9	7	9	7	9	9	4	7	9	4	9	7	9	7	9	9	4	7	9	4

(c)

Fig. 5

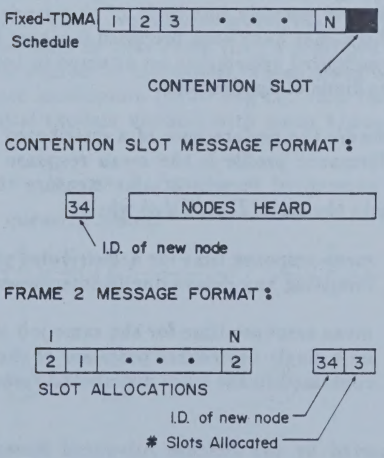


Fig. 6

ON THE THEORY OF DISTRIBUTED PROCESSING*

LEONARD KLEINROCK
University of California, Los Angeles

ABSTRACT

We consider a distributed processing environment in which a total processing capacity is split into smaller processing units (of the same total capacity), which collectively process a stream of jobs. We study the performance ratio T (the mean response time seen by jobs in this distributed environment) with T_0 (the mean response time seen by a job when it is processed in a centralized environment by a single processor). The most general configuration studied is that of a series-parallel topology. In particular, we consider m parallel chains, the kth of which contains n_k processors in series, each of capacity C_k operations/second. We assume that jobs arrive at the kth chain from a Poisson source at rate λ_k jobs/second and that each job requires an exponentially distributed number of operations from each processor. We find, for the symmetric system ($n_k = n$, $\lambda_k = \lambda/m$), that $T(\rho)/T_0(\rho) = mn$. For the general system (arbitrary n_k) but with equal loading on each series chain, we show that $T(\rho)/T_0(\rho) = \sum_1^m n_k$.

We find the optimal distribution of traffic among the chains; one property of this solution is that some of the series chains carry zero traffic. When we optimize the capacity assignment, we find that $\min n_k \leq T(\rho)/T_0(\rho) \leq \sum n_k$. When we do the joint optimization, we find that $T(\rho)/T_0(\rho) = \min n_k$. Distributed processing increases the mean response time! Lastly, we discuss the effect of processor costs on system performance.

1. Introduction

We have seen a steady progression from 8-bit processors to 16-bit processors and, more recently, to 32-bit processors. The price/performance profile has been improving dramatically due to the unprecedented revolution in the integrated chip technology. As a result, computer system designers are now considering distributed processing configurations with hundreds, thousands and possibly hundreds of thousands of processors. Many interconnection and structural schemes have been proposed (ranging from precisely planned iterative arrays to randomly configured systems) in an attempt to harness the combined power of these processors in various kinds of applications.

How are we to measure the performance of a distributed processing system? Perhaps the most traditional performance profile is the mean response time (T) versus the system throughput (ρ). It is a normalized version of this measure that we adopt in this paper. Specifically, we will evaluate the ratio $T(\rho)/T_0(\rho)$ where

$T(\rho) =$ mean response time for a *distributed* processing configuration (omitting any delays due to interprocessor communication),

and

$T_0(\rho) =$ mean response time for the same job stream when it is processed by a single *centralized* processor of the same capacity as that contained in the total distributed system.

*This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-82-C-0064.

One would hope that the advantage of distributed processing would manifest itself by producing a ratio such that $T(\rho)/T_0(\rho) \ll 1$. For the model developed below, we show that this is definitely *not* the case and, in fact, $T(\rho)/T_0(\rho) \geq 1$. The key assumption which leads to this (apparently negative) result is that the collection of processors in the distributed system has the *same total capacity* as that of the single processor to which it is being compared. What this comparison ignores is the relative *cost* of "small" versus "large" processors. Specifically, it is currently true that a 10 MIP (millions of instructions per second) processor costs far more than ten times that of a 1 MIP processor. Thus, a more meaningful comparison would be to compare a distributed system to a centralized system of the *same cost*. We approach that problem in Section 5 below. Beyond response time, throughput and cost, there are other (more intangible) performance measures one should really introduce in any such evaluation. For example, one should consider system reliability, flexibility, availability, simplified climate control, user friendliness, etc. as factors favoring distributed systems; on the negative side, one might include increased memory requirements, inter-processor communication, etc. We do not consider these features in this paper.

2. The Model

We consider a distributed processing environment in which a total processing capacity, C (operations/second), is split into a number of smaller processing units (of the same total capacity), which collectively process a stream of jobs.

The most general configuration studied in this paper is that of a series-parallel topology as shown in Figure 2.1 below. In particular, we consider m parallel chains, the kth of which contains n_k processors in series, each of capacity C_k operations/second, such that

$$C = \sum_{k=1}^{m} n_k C_k \tag{2.1}$$

(and, at no loss of generality, we assume that $C_1 \geq C_2 \geq \cdots \geq C_m$). We assume that jobs arrive at the kth chain from a Poisson source at rate λ_k jobs/second such that

$$\lambda = \sum_{k=1}^{m} \lambda_k \tag{2.2}$$

When a job passes down the kth chain, we assume it receives $1/\mu n_k$ operations, on the average, from each of the n_k processors in that chain (i.e., a pipelined processor configuration); thus, each job receives a total number of operations, whose mean is $1/\mu$ operations/job. We further make the independence assumption [KLEI 64], i.e., that the service received at each processor is an i.i.d. exponential random variable with mean $1/\mu n_k C_k$ seconds. In all cases, we assume that

$$\rho_k = \lambda_k/\mu n_k C_k < 1 \tag{2.3}$$

so, each processor is a stable queueing system.

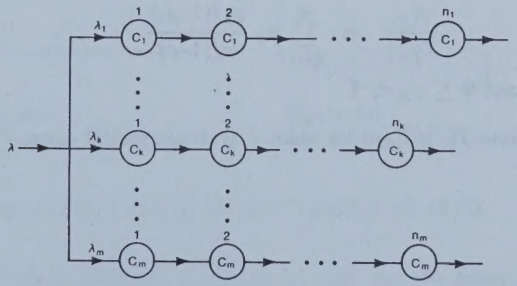


Figure 2.1: The General Series-Parallel Topology

3. Performance Evaluation

For the centralized processor, we have an M/M/1 queueing system for which the mean response time is [KLEI 75, Eq.(3.26)]

$$T_0(\rho) = \frac{\overline{x}}{1-\rho} \tag{3.1}$$

where $\overline{x}$ = E [service time] = $\frac{1}{\mu C}$ and $\rho = \lambda\overline{x} = \frac{\lambda}{\mu C}$. Thus,

$$T_0(\rho) = \frac{1}{\mu C - \lambda} \tag{3.2}$$

For the distributed system shown in Figure 2.1, we have [KLEI 76, Eq.(5.17)]

$$T(\rho) = \sum_{k=1}^{m} \frac{\lambda_k}{\lambda} T_k \tag{3.3}$$

where T_k = E [response time to pass down the k^{th} parallel chain]. Since the k^{th} chain contains n_k processors in series, we have

$$T_k = n_k T^{(k)} \tag{3.4}$$

where $T^{(k)}$ is the mean response time for *each* processor in the kth branch. Each of these processors behaves as an M/M/1 queue with input rate λ_k, mean number of operations $1/\mu n_k$ and capacity C_k operations/second. Thus, $T^{(k)}$ has the same form as given in Eq.(3.1) with $\overline{x}=1/\mu n_k C_k$ and $\rho=\lambda_k\overline{x}$. Thus,

$$T^{(k)} = \frac{1/\mu n_k C_k}{1 - \lambda_k/\mu n_k C_k} = \frac{1}{\mu n_k C_k - \lambda_k}$$

From this last equation and Eqs.(3.3) and (3.4), we get

$$T(\rho) = \sum_{k=1}^{m} \frac{\lambda_k}{\lambda} \cdot \frac{n_k}{\mu n_k C_k - \lambda_k} \tag{3.5}$$

This is the governing behavior of our general series-parallel topology.

Finally, we have the general form of our performance measure:

$$\frac{T(\rho)}{T_0(\rho)} = \sum_{k=1}^{m} \frac{\lambda_k n_k}{\lambda} \cdot \frac{\mu C - \lambda}{\mu n_k C_k - \lambda_k} \tag{3.6}$$

This last may also be rewritten as

$$\frac{T(\rho)}{T_0(\rho)} = \sum_{k=1}^{m} n_k \frac{\rho_k/(1-\rho_k)}{\rho/(1-\rho)} \tag{3.7}$$

where $\rho_k = \lambda_k/\mu n_k C_k$ and $0 \leq \rho_k < 1$.

Let us now evaluate $T(\rho)/T_0(\rho)$ for some interesting, special cases:

THE PURE TANDEM

This system is depicted in Figure 3.1 and has the following parameter values: $m=1$, $n_1=n$, $\lambda_1=\lambda$, $1/\mu_1=1/n\mu$ and $C_1=C/n$. Substituting into Eq.(3.6), we have

$$\frac{T(\rho)}{T_0(\rho)} = \frac{\lambda n}{\lambda} \cdot \frac{\mu C-\lambda}{\mu n \frac{C}{n}-\lambda} = n \tag{3.8}$$

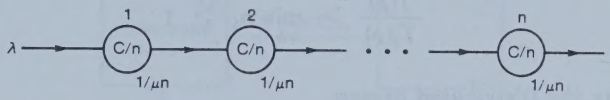


Figure 3.1: The Pure Tandem System

THE PURE PARALLEL SYSTEM

This system is depicted in Figure 3.2 and has the following parameter values: $n_k=1$, $\lambda_k=\lambda/m$, $1/\mu_k=1/\mu$, $C_k=C/m$ for $k=1,2,\ldots,m$. Substituting into Eq.(3.6), we have

$$\frac{T(\rho)}{T_0(\rho)} = \sum_{k=1}^{m} \frac{\lambda/m}{\lambda} \frac{\mu C-\lambda}{\mu \frac{C}{m}-\frac{\lambda}{m}} = m \tag{3.9}$$

THE SYMMETRIC SERIES-PARALLEL SYSTEM

This system is depicted in Figure 3.3 and has the following parameter values: $n_k=n$, $\lambda_k=\lambda/m$, $1/\mu_k=1/n\mu$, $C_k=C/mn$ for $k=1,2,\ldots,m$. Substituting into Eq.(3.6), we have

$$\frac{T(\rho)}{T_0(\rho)} = \sum_{k=1}^{m} \frac{\lambda n/m}{\lambda} \cdot \frac{\mu C-\lambda}{\frac{\mu n C}{mn}-\frac{\lambda}{m}} = nm \tag{3.10}$$

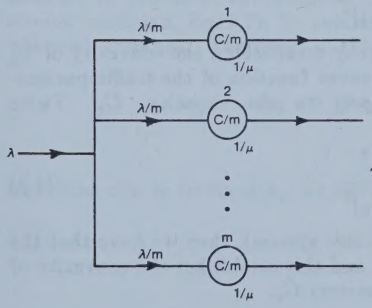


Figure 3.2: The Pure Parallel System

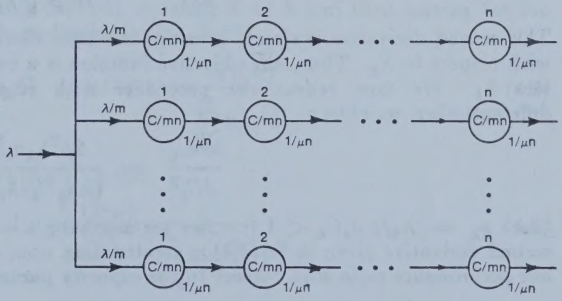


Figure 3.3: The Symmetric Series-Parallel System

THE SERIES-PARALLEL SYSTEM WITH UNIFORM TRAFFIC

This system is depicted in Figure 2.1 and has the following parameter values: $\lambda_k=\lambda/m$, $1/\mu_k=1/n_k\,\mu$, $C_k=C/mn_k$ for $k=1,2,\ldots,m$. Substituting into Eq.(3.6), we have

$$\frac{T(\rho)}{T_0(\rho)} = \sum_{k=1}^{m} \frac{\lambda n_k/m}{\lambda} \cdot \frac{\mu C - \lambda}{\mu \frac{C}{m} - \frac{\lambda}{m}} = \sum_{k=1}^{m} n_k \tag{3.11}$$

Thus, we see, for *all* of the simple configurations, that things seem to be getting *worse* as we add more distributed processors! Indeed, we show below that, no matter how one selects λ_k and C_k, subject to constraint Eqs.(2.1, 2.2 and 2.3), one must have

$$\frac{T(\rho)}{T_0(\rho)} \geq \min_k n_k \geq 1 \tag{3.12}$$

4. Optimizing the Distributed System

In the previous section, we evaluated the behavior of the general series-parallel system. In this section, we consider some *optimization* problems, where we allow ourselves to adjust λ_k and/or C_k (subject to constraints 2.1, 2.2 and 2.3) in order to minimize $T(\rho)/T_0(\rho)$.

Before we proceed with these optimization problems, let us establish certain convexity properties of our objective function Eq.(3.6), which we repeat here.

$$\frac{T(\rho)}{T_0(\rho)} = \sum_{k=1}^{m} \frac{\lambda_k n_k}{\lambda} \cdot \frac{\mu C - \lambda}{\mu n_k C_k - \lambda_k} = \frac{\mu C - \lambda}{\lambda} \sum_{k=1}^{m} U_k \tag{4.1}$$

where

$$U_k = \frac{\lambda_k n_k}{\mu n_k C_k - \lambda_k} \tag{4.2}$$

We shall now prove that U_k is a convex function of its various arguments; then, since our performance ratio as given in Eq.(4.1) is a sum of the terms U_k, we shall have established that the performance ratio itself is a convex function of the system parameters (a sum of convex functions is convex). First we prove that U_k is a convex function of the traffic parameter λ_k by twice differentiating, namely,

$$\frac{d^2 U_k}{d\lambda_k^2} = \frac{2\mu n_k^2 C_k}{(\mu n_k C_k - \lambda_k)^2} \tag{4.3}$$

This second derivative is clearly greater than zero, thereby establishing the convexity of U_k with respect to λ_k. Thus, our objective function is a convex function of the traffic parameters λ_k. We now repeat the procedure with respect to the capacity, C_k. Twice differentiating, we obtain

$$\frac{d^2 U_k}{dC_k^2} = \frac{2\mu^2 \lambda_k n_k^3}{(\mu n_k C_k - \lambda_k)^3} \tag{4.4}$$

Since $\rho_k = \lambda_k/\mu n_k C_k < 1$ (i.e., we are assuming a stable system), then we have that the second derivative given in Eq.(4.4) is greater than zero, and this establishes the convexity of our performance ratio with respect to the capacity parameters C_k.

The set of optimization problems we wish to address may be identified by referring to Figure 4.1. In that figure, box A refers to a system with an arbitrary set of parameters for which no optimization has been performed; thus, the performance of box A is given by Eq.(3.6).

The first problem we address is to hold C_k and n_k fixed and optimally select the traffic pattern, λ_k, in order to minimize our performance ratio $T(\rho)/T_0(\rho)$. This leads us to box B. The second problem we solve is to hold λ_k and n_k fixed and optimally assign the total channel capacity C among the set of distributed processors, once again minimizing the performance ratio $T(\rho)/T_0(\rho)$, which leads us to box C. To reach box D, which is the joint optimization of both λ_k and C_k, we may either move from box B by optimizing the capacity or move from box C by optimizing the traffic. As we show below, both solutions are identical.

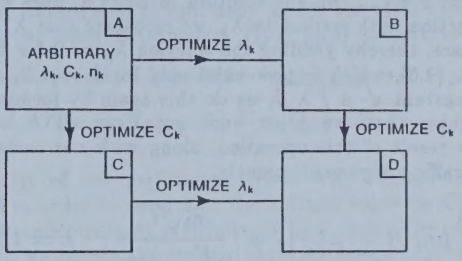


Figure 4.1

We begin with the problem characterized in box B, namely,

$$\underset{\lambda_k}{Minimize} \quad \frac{T(\rho)}{T_0(\rho)}$$

subject to Eq.(2.2) for a given set C_k, n_k and μ. Clearly, this is equivalent to minimizing $T(\rho)$ since $T_0(\rho)$ is a constant for this problem. Thus, we set up the Lagrangian

$$G = T(\rho) + \beta \left(\sum_{k=1}^{m} \lambda_k - \lambda \right) \tag{4.5}$$

where β is the Lagrangian undetermined multiplier. Forming the partial derivative of G with respect to λ_k, setting it equal to zero for each k and solving for λ_k, we obtain

$$\lambda_k = \mu n_k C_k - \sqrt{-\mu / \lambda \beta}\, n_k \sqrt{C_k} \tag{4.6}$$

In order to determine the unknown constant, namely, $\sqrt{-\mu / \lambda \beta}$, we simply form the constraint condition, Eq.(2.2), by summing Eq.(4.6), equating it to λ and then solving for the unknown constant. The result of this evaluation yields, finally,

$$\lambda_k = \mu n_k C_k - \mu C (1-\rho) \frac{n_k \sqrt{C_k}}{\sum_{i=1}^{m} n_i \sqrt{C_i}} \tag{4.7}$$

Rewriting this in terms of ρ_k, we have (for $C_k > 0$)

$$\rho_k = 1 - \frac{C(1-\rho)}{\sqrt{C_k}} \cdot \frac{1}{\sum_{i=1}^{m} n_i \sqrt{C_i}} \tag{4.8}$$

Either Eq.(4.7) or (4.8) gives the solution for the optimal assignment of traffic to the generalized series-parallel distributed processing network. However, we observe that the constraints on ρ are $0 \leq \rho_k < 1$. Clearly, the upper constraint is satisfied as long as $\rho < 1$, a condition we have assumed. It is the lower constraint which can be violated if, indeed, the capacity, C_k, is too small. That is, we must take care that our optimization problem does not try to place a *negative* flow λ_k on the kth parallel chain! Since we know that the objective function is a convex function of the flows, then when λ_k tries to go negative, the optimization procedure will force us to keep it at its boundary value of zero. Thus, the solution for λ_k given in Eq.(4.7) will hold only if the following condition is satisfied (C^*, ρ^* and k^* defined below):

$$\sqrt{C_{k^*}} \geq \frac{C^*(1-\rho^*)}{\sum_{i=1}^{k^*} n_i\sqrt{C_i}} \tag{4.9}$$

Any chain k for which condition (4.9) is not satisfied will lead to an optimum flow of value $\lambda_k = 0$. Since we have assumed that the capacities, C_k, are monotonically decreasing with the index k, then we may define k^* as the maximum value of k for which condition (4.9) holds. For $k = k^*+1,\ldots, m$, the solution in Eq.(4.6) does not apply, but due to the convexity of our function with respect to λ_k, we recognize that λ_k must lie at the boundary of our constraint space, thereby yielding the solution $\lambda_k = 0$ for $k = k^*+1,\ldots, m$. Returning once again to Eq.(4.6), which is now valid only for $k = 1,2,\ldots,k^*$, we must re-evaluate the undetermined constant $\sqrt{-\mu / \lambda \beta}$; we do this again by forming the constraint Eq.(2.2), but we now recognize that we must sum not over all k but only over the range $k = 1,2,\ldots,k^*$. The result of this operation, along with our earlier observation, yields the final result for our traffic assignment, namely,

$$\lambda_k = \begin{cases} \mu n_k C_k - \mu C^*(1-\rho^*)\dfrac{n_k\sqrt{C_k}}{\sum_{i=1}^{k^*} n_i\sqrt{C_i}} & k = 1,2,\ldots,k^* \\ 0 & k = k^*+1, \ldots, m \end{cases} \tag{4.10}$$

where we have defined the following quantities

$$C^* = \sum_{k=1}^{k^*} n_k C_k \tag{4.11}$$

$$\rho^* = \lambda/\mu C^* \tag{4.12}$$

In order to determine the optimum performance ratio, we now substitute our optimized traffic assignment as given in Eq.(4.10) into the performance equation, Eq.(3.6), to yield

$$\left.\frac{T(\rho)}{T_0(\rho)}\right|_{\text{opt } \lambda_k} = \frac{\mu\rho^*(1-\rho)}{\lambda\rho(1-\rho^*)}\left[\sum_{k=1}^{k^*} n_k\sqrt{C_k}\right]^2 - \frac{N^*(1-\rho)}{\rho} \tag{4.13}$$

where we have defined N^* to be the total number of processors in our distributed system which carry a non-zero traffic, namely,

$$N^* = \sum_{k=1}^{k^*} n_k \tag{4.14}$$

Eq.(4.13) is the solution to the optimization problem represented by box B.

The optimization problem we just solved is, clearly, the solution to the Flow Assignment (FA) problem described in [KLEI 76], except that here we have a special topology to work with; so, we are able to develop an explicit analytic solution rather than the usual algorithmic solution (e.g., the Flow Deviation (FD) method in [KLEI 76, Sect.5.8]). We observe that $l^{(k)}$, the "length" (defined for the FD method) of the kth parallel chain is

$$l^{(k)} = \begin{cases} \dfrac{1}{\lambda}\left[\dfrac{\sum_{i=1}^{k^*} n_i\sqrt{C_i}}{C^*(1-\rho^*)}\right]^2 & k = 1,2,\ldots,k^* \\ \dfrac{1}{\lambda\mu C_k} & k = k^*+1, \ldots, m \end{cases} \tag{4.15}$$

It can be seen from Eq.(4.9) that $l^{(k)}$ for $k = k^*+1,\ldots, m$ is strictly greater than $l^{(k)}$ for $k = 1,2,\ldots,k^*$. Thus, each of the zero-traffic parallel chains ($k>k^*$) is *longer* than each of the positive-traffic chains ($k\leq k^*$), which (by the FD method) implies that no traffic should be deviated onto the zero-traffic chains. Further, since each of the positive-traffic chains has the same "length," then no traffic should be deviated among those chains. Thus, our solution is in complete agreement with the FD method.

Now, let us proceed from box A to box C, namely, let us assume that the traffic λ_k remains fixed and that our objective is to optimally assign the total capacity, C, among the set of processors, that is, we must $\underset{C_k}{Minimize}$ $T(\rho)/T_0(\rho)$ subject to constraint Eq.(2.1) and, of course Eq.(2.3). The appropriate Lagrangian in this case is

$$G = T(\rho) + \beta\left(\sum_{k=1}^{m} n_k C_k - C\right) \tag{4.16}$$

Again, due to the convexity of our performance ratio with respect to C_k, we may form $\partial G / \partial C_k = 0$ for all k in order to determine the optimal value for C_k. Carrying out this differential, evaluating the undetermined multiplier β by forming the constraint as given in Eq.(2.1) and solving for C_k, we obtain the optimal capacity assignment as

$$C_k = \frac{\lambda_k}{\mu n_k} + \frac{C(1-\rho)\sqrt{\lambda_k/n_k}}{\sum_{i=1}^{m}\sqrt{\lambda_i n_i}} \tag{4.17}$$

As long as $\rho < 1$, then the optimal capacity assignment given in Eq.(4.17) always holds, as opposed to the traffic assignment problem where we encountered negative flows. If we now plug this optimized capacity assignment into our performance ratio as given in Eq.(3.6), we then obtain the minimized response time ratio, namely,

$$\left.\frac{T(\rho)}{T_0(\rho)}\right|_{\text{opt } C_k} = \left(\sum_{k=1}^{m}\sqrt{\lambda_k n_k/\lambda}\right)^2 \tag{4.18}$$

Note the similarity of this capacity optimization procedure and solution to that given in Eq.(5.28) of [KLEI 76]. The optimum performance profile for box C is that given in Eq.(4.18).

Let us now further optimize the solution in Eq.(4.18) with respect to the traffic flows λ_k, namely,

$$\underset{\lambda_k}{Minimize} \quad \left.\frac{T(\rho)}{T_0(\rho)}\right|_{\text{opt } C_k} \tag{4.19}$$

The solution procedure here is almost identical to that given in Exercise 5.5 of [KLEI 76] and leads to the following traffic assignment:

$$\lambda_k = \begin{cases} \lambda & k=k_0 \\ 0 & k\neq k_0 \end{cases} \tag{4.20}$$

where k_0 is such that $n_{k_0} \leq n_k$ for all k; in the case where more than one value of k qualify for the value k_0, then selecting any one of them will yield the same optimum solution. We are assuming, in this double optimization problem, that each time the λ_k assignment changes, then the optimized capacity assignment follows that change (i.e., through Eq.(4.17); thus, the solution to problem (4.19) will yield a traffic pattern where all the traffic travels down the minimum length parallel chain, and that chain will contain the total capacity, C, uniformly spread out over n_{k_0} processors. In this case, we have now arrived at a pure tandem, whose solution is known from Eq.(3.8); so, the performance ratio which applies to box D, arrived at through this capacity and then traffic assignment optimization (i.e., box A to box C to box D), gives

$$\left.\frac{T(\rho)}{T_0(\rho)}\right|_{\substack{\text{opt}\\ C_k,\lambda_k}} = n_{k_0} \tag{4.21}$$

On the other hand, we could have begun with the solution from box B and then done an optimum capacity assignment to arrive at box D, a procedure which we now describe. Thus, we carry out the following minimization

$$\underset{C_k}{\textit{Minimize}} \quad \left.\frac{T(\rho)}{T_0(\rho)}\right|_{\text{opt } \lambda_k} \tag{4.22}$$

Of course, this result must lead to a performance ratio as given in Eq.(4.21). The approach here is not quite as simple as going from box C to box D. In particular, the problem comes about from the way in which we progressed from Eq.(4.7) to Eq.(4.10), namely, we required $\lambda_k = 0$ for $k = k^*+1, \ldots, m$ in order to prevent those values of λ_k from going negative. However, now that we allow ourselves to optimize the capacity to match the traffic flow (i.e., we will move from box B to box D), we may take the liberty of ignoring this non-negativity constraint, optimizing the capacity and then checking, after this capacity optimization, to see that the non-negativity constraint is satisfied by the λ_k's. Thus, for a given set, C_k, and chain lengths, n_k, we accept that the optimum traffic assignment is given by Eq.(4.7) for all k. Given this assignment for λ_k, we will now move from box B to box D by optimizing the capacity relative to the λ_k found from Eq.(4.7). But, of course, we have already carried out the capacity optimization in moving from box A to box C, and the solution is given in Eq.(4.17). Let us evaluate this optimized capacity assignment by substituting λ_k from Eq.(4.7) into our expression for C_k in Eq.(4.17), namely,

$$C_k = C_k - \frac{C(1-\rho)\sqrt{C_k}}{\sum_{i=1}^{m} n_i\sqrt{C_i}} + \frac{C(1-\rho)y_k}{\sum_{i=1}^{m} y_i n_i} \tag{4.23}$$

where

$$y_k = \left(\frac{\lambda_k}{n_k}\right)^{1/2} = \left(\mu C_k - \frac{\mu C(1-\rho)\sqrt{C_k}}{\sum_{i=1}^{m} n_i\sqrt{C_i}}\right)^{1/2} \tag{4.24}$$

Cancelling C_k on both sides of Eq.(4.23) and solving, we obtain

$$\sqrt{C_k} = \left(\sum_{i=1}^{m} n_i\sqrt{C_i}\right)\frac{y_k}{\sum_{i=1}^{m} y_i n_i} \tag{4.25}$$

Observing that y_k is a function of C_k, what are the solutions to this set of equations ($k = 1,2,\ldots,m$)? It can be shown that there are exactly m solution sets to this equation, each of the form

$$C_k = \begin{cases} C/n_{k'} & k=k' \\ 0 & k\neq k' \end{cases} \tag{4.26}$$

where the m solutions correspond to the m possible values for k', namely, $k' = 1,2,\ldots,m$. Plugging any one of these solutions for C_k into Eq.(4.24), we find

$$y_k = \begin{cases} \sqrt{\lambda/n_{k'}} & k=k' \\ 0 & k\neq k' \end{cases}$$

It is now easy to verify that Eq.(4.26) satisfies Eq.(4.25). Each of these m solutions corresponds to a pure tandem, for which the performance ratio is given in Eq.(3.8), namely,

$$\frac{T(\rho)}{T_0(\rho)} = n_{k'} \tag{4.27}$$

Among the m possible solutions, that which minimizes this performance ratio is clearly the case $k' = k_0$ where, we recall, k_0 is defined as that chain with the minimum number of series processors, namely, $n_{k_0} \leq n_k$ for all k. This then yields the optimum performance profile for box D (by travelling from box A to box B to box D), namely,

$$\left.\frac{T(\rho)}{T_0(\rho)}\right|_{\text{opt } \lambda_k, C_k} = n_{k_0} \tag{4.28}$$

Thus, we see that the double optimization, in both cases, has led to a solution where all the traffic and all the capacity are placed on that chain with the fewest number of nodes in series. We now observe that the solution in box B allows us to put tight upper and lower bounds on our performance profile for any capacity-optimized series-parallel system. The general form for $T(\rho)/T_0(\rho)$ is given in Eq.(4.18). If we let $a_k = \sqrt{\lambda_k/\lambda}$ and $b_k = \sqrt{n_k}$, then we may apply the Cauchy-Schwartz inequality (see Eq.(4.59) in [KLEI 75]), namely,

$$\left(\sum_{k=1}^{m} a_k\, b_k\right)^2 \leq \sum_{k=1}^{m} a_k^2 \sum_{k-1}^{m} b_k^2 \tag{4.29}$$

to obtain

$$\left.\frac{T(\rho)}{T_0(\rho)}\right|_{\text{opt } C_k} \leq \sum_{k=1}^{m} n_k \tag{4.30}$$

Furthermore, since $n_k \geq n_{k_0}$, then $(\sum\sqrt{\lambda_k\, n_k/\lambda})^2 \geq n_{k_0}(\sum\sqrt{\lambda_k/\lambda})^2$. Now, since $\lambda_k/\lambda \leq 1$, it follows that $\sqrt{\lambda_k/\lambda} \geq \lambda_k/\lambda$, and so, $(\sum\sqrt{\lambda_k/\lambda})^2 \geq (\sum\lambda_k/\lambda)^2 = 1$. Thus we have shown that Eq.(4.18) is lower bounded by n_{k_0}, that is,

$$n_{k_0} \leq \left.\frac{T(\rho)}{T_0(\rho)}\right|_{\text{opt } C_k} \leq \sum_{k=1}^{m} n_k \tag{4.31}$$

From Eq.(4.28), we see that this lower bound is, indeed, achieved in box D. (Note also that the upper bound was achieved in the series-parallel system with uniform traffic (Eq.(3.11)).

If one had the further freedom to select the length of this minimum-length chain, then it is clear that one would select $n_{k_0} = 1$ to obtain the absolutely optimum performance, which corresponds to our centralized system! Thus, we see that, so far as *this* performance profile is concerned, distributed processing can only hurt us.

5. Cost Considerations

As promised, we now return to certain issues involving cost. In particular, we were disappointed to find that distributed processing offers no advantage as far as our performance ratio is concerned, and we argued earlier that this negative result is due largely to the assumption that the single, centralized processor of capacity C costs the same as a distributed system whose total capacity, C_{dist}, is equal to C. However, as we pointed out earlier, there is, basically, a dis-economy in scale for the cost of processor capacity due to the integrated chip revolution, and it is this fact that has the potential for "saving" distributed processing so far as our model is concerned. In order to introduce this factor into our analysis, we will find how much more capacity we must place in the distributed system relative to the centralized system in order that $T(\rho)/T_0(\rho) = 1$; i.e., we shall adjust C_{dist} so that the distributed system and the centralized system give the same mean response times.

Let us consider the symmetric series-parallel system, introducing the two different capacities. Re-deriving Eq.(3.11), we find

$$\frac{T(\rho)}{T_0(\rho)} = mn\frac{\mu C-\lambda}{\mu C_{dist}-\lambda} \tag{4.32}$$

Forcing this ratio to be unity, we find that the ratio of capacities in the two systems must be

$$\frac{C_{dist}}{C} = mn-\rho(mn-1) \tag{4.33}$$

where, once again, $\rho = \lambda/\mu C$. We note that this ratio of capacities is dependent upon the centralized system's load, ρ. In particular, we notice that $\lim_{\rho\to 0} C_{dist}/C = mn$ and also $\lim_{\rho\to 1} C_{dist}/C = 1$; i.e., at light load we require an increase in capacity equal to the number of processors in the distributed system, whereas at heavy load we need only an infinitesimal extra capacity (due to the steep slope of the mean response time near saturation).

6. Conclusions

We focussed on the ratio of response time of a distributed system to that of a centralized system as our performance measure. We found that distributed processing can never improve that ratio unless we introduce a dis-economy of scale in the cost of processing. We solved a number of related optimization problems, finding the optimum traffic assignment and the optimum capacity assignment, and when both those assignments are optimized jointly, we were led back to the centralized system. A number of extensions to this model are currently being investigated.

REFERENCES

[KLEI 64] Kleinrock, L., *Communication Nets; Stochastic Message Flow and Delay,* McGraw-Hill Book Company, New York, 1964. (Out of Print) Reprinted by Dover Publications, 1972.

[KLEI 75] Kleinrock, L., *Queueing Systems, Volume I: Theory,* Wiley Interscience, New York, 1975.

[KLEI 76] Kleinrock, L., *Queueing Systems, Volume II: Computer Applications,* Wiley Interscience, New York, 1976.

Criticality Considerations in the Design of Geometric Algorithms

Bernard Chazelle
Department of Computer Science
Brown University
Providence, RI 02912

Abstract

We examine a class of geometric problems which all share a feature of considerable relevance: the underlying set is defined implicitly and is at least *quadratically* larger than the input size. Is efficient computation possible without expanding the set explicitly? If not, are non-trivial lower bounds provable? We give partial answers to these questions by proposing various approaches for attacking these problems. In doing so, we improve a number of complexity bounds for problems of range searching, frame-to-frame coherence, segment stabbing, geometric predicate computation, etc.

1. Triangular Search

Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in E^2. The triangular search problem is to preprocess S so that, for any query triangle T, the subset of points of S lying inside T can be computed efficiently. An $(f(n), g(n))$ solution refers to an algorithm for this problem that requires $O(f(n))$ space and $O(g(n)+$ output size) query time. The most efficient solutions to date are an $(n, n^{.695})$ algorithm by Edelsbrunner and Welzl [EW] and an $(n^{2+\epsilon}, \log n \log \frac{1}{\epsilon})$ by Cole and Yap [CY] (see also [EKM,W] for other methods). Our first observation is that quadratic space complexity "seems" to be the price to pay for fast retrieval. This can be understood by looking at the problem of determining whether a query line L passes through any point of S. Obviously, this is a subproblem of the original problem: simply make the triangle T infinitely long and skinny. Searching is (at least conceptually) facilitated by turning to a dual space, whereby lines becomes points and points become lines. The dual of S is an arrangement of n

lines, so the subproblem is reduced to locating the dual point of L in its $O(n^2)$ possible (topologically distinct) locations. This suggests a straightforward $O(n^2, \log n)$ solution via planar point location, but also points to the difficulty of escaping this costly approach. Cole and Yap's solution to triangular search can be understood in this light: its high space complexity comes from its enforced discrimination among *critical* positions. We propose to show that interestingly the subproblem where T becomes a skinny elongated triangle (going towards a line) is not just *some* hard subproblem: it is the *only* hard subproblem. We will prove that an efficient solution always exists provided that T is not exceedingly lean. Specifically, we require that each angle of T should be bounded below by a constant α.

Theorem 1. There exists an $(n \log n, \log^2 n)$ solution to the bounded-angle triangular search problem.

Proof: We need two preliminary results: to begin, we recall a result shown in [CG]. Given a set of S of n points, there exists an $O(n)$ data structure, denoted $T_L(S)$, s.t. for any query grounded right triangle q, the set $S \bigcap q$ can be computed in time $O(\log n + |S \bigcap q|)$ (a right triangle is grounded if one of its non-hypothenuse edges lies on a fixed line — the result in [CG] actually also applies to trapezoids, but this is not needed here). Consider now the original problem, with the added restriction that T be a right triangle with one of its non-hypothenuse edges parallel to a fixed line L. Wlog assume that no two points in S form a segment parallel to L. Let $\{p_1, \ldots, p_n\}$ be the points of S sorted according to the order of their projections on a normal to L. Let $L_0, L_1, \ldots, L_n$ be a set of lines parallel to L with p_i being the only point to lie between L_{i-1} and L_i. Let $\mathcal{T}$ be the complete binary tree with its left-to-right leaf sequence in one-to-one correspondence with $\{p_1, \ldots, p_n\}$. Note that each node v of $\mathcal{T}$ spans a subset $S(v)$ of S between some $L'(v) = L_i$ and $L''(v) = L_j$. For each node $v \in \mathcal{T}$, compute the two structures $T_{L'(v)}(S(v))$ and $T_{L''(v)}(S(v))$. This provides us with an $(n \log n, \log^2 n)$ solution for the restricted problem via a trivial canonical decomposition of the query triangle T into $O(\log n)$ (overlapping) triangles.

Next we turn to the general problem. Assume that each angle of T exceeds α. Let d_i be the unit vector with $\angle(\vec{x}, \vec{d_i}) = i\alpha$, defined for each i; $0 \leq i \leq \lceil \frac{2\pi}{\alpha} \rceil - 1$. We set up a data structure of the previous type for each direction in $\Delta = \{d_0, \ldots, d_{\lceil \frac{2\pi}{\alpha} \rceil - 1}\}$. The theorem will trivially follow once we show how to partition T into a constant number (≤ 12) of right triangles with

one (non-hypothenuse) side parallel to some $d \in \Delta$. Let ABC be an arbitrary triangle, each of whose angles exceeds α. We can always assume wlog that each angle is at most $\pi/2$. If this is not the case, split T in two by drawing the angle bisector from the unique obtuse-angled vertex. With these conditions satisfied, there must exist a segment AA' with direction D_a and $A' \in BC$ (Fig.1). Let I_B (resp. I_C) be the intersection of AA' with the normal to AA' passing through B (resp. C). Since $\angle(AB, AC) \leq \frac{\pi}{2}$, T lies entirely on one side of the normal to AA' passing through A, therefore we have $I_B \in AI_C$ or $I_C \in AI_B$. Wlog assume the former case. Let B' be the intersection of AC with the line passing through BI_B. Similarly, there exists a segment CC' with direction D_c and $C' \in BB'$. Since AI_BB' forms a right triangle, $\angle(B'B, B'C)$ is obtuse, therefore the normal to CC' passing through B' intersects CC' at some point, denoted $J_{B'}$. Let D be the intersection of BC with the line passing through $B'J_{B'}$. Since $CJ_{B'}D$ forms a right triangle, $\angle(DB', DB)$ is obtuse, so the normal to BB' passing through D intersects BB' at some point $D' \in BB'$. This completes the partitioning of ABC into six right triangles $\{AI_BB, AI_BB', BD'D, B'D'D, DJ_{B'}C, B'J_{B'}C\}$, each with one (non-hypothenuse) side parallel to a direction in Δ. ∎

This example illustrates the *buffering* effect of "filtering search" [C1], whereby reporting many points seems relatively as easy as reporting only one. The notion of efficiency for retrieval problems is traditionally captured by complexities of the form $(nPOLYLOG(n), POLYLOG(n))$. Whether triangular search is in this class, i.e. whether the quadratic complexity of the underlying criticality set can be overcome, remains open. What we have shown is that we can restrict future investigation to the case where T is arbitrarily close in shape to a segment.

2. Frame-to-Frame Coherence

Hidden-line elimination problems are many, varied, and often quite difficult. One reason for the latter is that once again the input is generally not amenable to searching, so one has to expand the input into a larger set where critical objects appear explicitly. The problem of maintaining frame-to-frame coherence in *Flatland* (i.e. in E^2) has been studied extensively by Edelsbrunner et al [EOW]. The difficulty of this restricted case of hidden-line elimination is well encapsulated in the following set of problems. Let $S = \{s_1, \ldots, s_n\}$ be a set of n segments in the plane and let C be a circle enclosing S. Let $\text{int}(A)$ denote the interior of figure A. We assume that $[\forall i, j; 1 \leq i \leq j \leq n \mid \text{int}(s_i) \bigcap \text{int}(s_j) = \emptyset]$. This will allow S, for example, to be

a set of simple polygons. Let (Ox, Oy) be an orthogonal system of coordinates with O at the center of C. For each θ $(0 \leq \theta < 2\pi)$, let $v(\theta)$ be the point of C s.t. $\angle(Ox, Ov(\theta)) = \theta$ and let $T(\theta)$ be the tangent to C at $v(\theta)$. Define $V(\theta)$ to be the parallel view of S from $T(\theta)$, i.e. the function $p \in T(\theta) \mapsto f(p) \in \{0, 1, \ldots, n\}$, where $s_{f(p)}$ contributes the nearest intersection to p between S and the normal to $T(\theta)$ at p (Fig.2). For consistency, $f(p) = 0$ if the intersection is empty. Also, let L be an oriented line and $H(L)$ be the index of the first segment of S to be intersected by L; once again, $H(L) = 0$ if no intersection.

P1. For any θ; $0 \leq \theta < 2\pi$, compute $V(\theta)$.

P2. For any θ; $0 \leq \theta < 2\pi$, and any segment $q \subseteq T(\theta)$, compute the restriction of $V(\theta)$ to the domain q.

P3. For any oriented line L, compute $H(L)$.

P4. Is it possible to see through S? More formally put, is there at least one value of θ s.t. 0 appears at least three times in the sequence $V(\theta)$?

P5. Is there a line L that has exactly k segments of S on one side and $n-k$ on the other side (s.t. $S \bigcap L = \emptyset$).

Theorems 2,3 improve the best previous results by factors of n space (1,2), $\log n$ space (3), and $\log n$ time (4,5).

Theorem 2. Problems 1,2,3 can be solved in $(n^2, \log n)$.

Proof: Let $\Omega = C \bigcap T(\theta) = v(\theta)$. We consider the local reference system $(\Omega\theta, \Omega z)$ where (θ, z) are the coordinates of the point p of $T(\theta)$ at a distance z from Ω. We resolve the ambiguity of this definition by stipulating that z is positive iff O, Ω, p form a left turn. The functions $V(\theta)$ are easily seen to subdivide the θz plane into $O(n^2)$ regions delimited by curves of the form $z = r\sin(\alpha - \theta)$ — Fig.3 depicts the subdivision corresponding to Fig.2. Each region is assigned an index i indicating that any ray of light emanating from a point in this region towards int(C) in a direction normal to $T(\theta)$ will first hit s_i. To construct the subdivision, denoted $\mathcal{T}$, can be done using a standard sweep-line technique. One will first sort the slopes of all inter-endpoint segments and compute $V(0)$. The precomputed slopes are taken as event-points of a sweep-line process updating $V(\theta)$ as θ grows from 0 to 2π — details omitted. The computation is easily done in $O(n^2 \log n)$ time and $O(n^2)$ space. The subdivision $\mathcal{T}$ is represented by a DCEL [MP]

or quad-edge structure [GS], with each edge (vertex) pointing to the one (two) segment(s) of S from which they originate. The interesting observation is that since each edge is monotone along the θ-axis, 1) $\mathcal{T}$ can be preprocessed for efficient point location as described in [EGS], 2) one can build up a *hive-graph* on the set of edges [C1]. Recall that the latter construction will require here $O(n^2 \log n)$ time, and will allow us to compute all the intersections between $\mathcal{T}$ and a query segment parallel to the z-axis in $(n^2, \log n)$. With this preprocessing, solving Problems 1-3 is easy: 1,2) - compute $\{\Theta - \theta = 0\} \bigcap \mathcal{T}$ (1) or $q \bigcap \mathcal{T}$ (2), using the hive-graph; 3) - compute $T(\theta)$, the first normal to L tangent to C traversed by L. Next locate $(\theta, z) = L \bigcap T(\theta)$ in $\mathcal{T}$ and report the corresponding index. ∎

Observation 1. Let $I = \{I_1, \ldots, I_n\}$ be a list of n closed intervals on a circle C, with all the endpoints stored in sorted order in a circular list. Let J be the set of intervals formed by $C - \left(\bigcup_{1 \leq i \leq n} I_i\right)$. If J is not empty, let $J_1, \ldots, J_p$ be its elements in sorted order around C, and let $K = C - \left(\bigcup_{1 \leq i \leq p} J_i\right)$. K consists of p intervals $K_1, \ldots, K_p$, with the convention that K_i immediately follows J_i in clockwise order. Note that $\bigcup_{1 \leq i \leq p}(J_i \cup K_i) = C$. Let k_i denote the number of intervals of I that make up K_i. In $O(n)$ time it is possible to determine whether J is empty, and if not, compute the sorted list $\{(J_1, k_1), \ldots, (J_p, k_p)\}$.

Proof: Choose a point O on C and, in $O(n)$ time, use this point to split in two every interval of I overlapping O. This allows us to order each endpoint v according to the length of the clockwise arc Ov. In $O(n)$ time, we express the augmented set of intervals as a list of the form $\{[a_1, b_1], \ldots, [a_m, b_m]\}$ $(n \leq m \leq 2n)$ with $b_1 \leq \ldots \leq b_m$. We compute J by considering each interval $[a_i, b_i]$ in turn. During the computation, the list of intervals currently in J is stored in a stack. For each $i = 1, \ldots, m$, process $[a_i, b_i]$ by popping off the stack all the intervals covered by $[a_i, b_i]$; update top of stack and iterate. To compute $\{k_1, \ldots, k_p\}$ is trivially done in $O(n)$ time by simulating a merge between the endpoints of J and those of I (taken here before splitting). ∎

Theorem 3. Problems 4,5 can be solved in $O(n^2)$ time.

Proof: Let $R(v, \theta)$ be the ray emanating from v with slope θ; $(0 \leq \theta < 2\pi)$. For any endpoint v in S, define $R(v) = \{\theta \in [0, 2\pi) \mid \forall s \in S, s \bigcap R(v, \theta) = \emptyset\}$. $R(v)$ consists of a (possibly empty) set of open intervals. This set corresponds to all the maximal wedges centered at v from which v can see to infinity. Suppose that $R(v)$ is not empty; then let $R(v) = \{J_1, \ldots, J_p\}$ in angular

order around v, and let k_i be the number of segments of S in the wedge between J_i and J_{i+1} (mod p). Let $L(v)$ be the list of endpoints in S (except v) sorted angularly around v. If $L(v)$ is given for each endpoint v in S, Observation 1 can be trivially adapted to compute the list $\{(J_1, k_1), \ldots, (J_p, k_p)\}$ in $O(n)$ time. By *sliding* the interval $[\theta, \theta + 2\pi]$ through $R(v)$ from $\theta = 0$ to $\theta = \pi$, it is then easy in $O(n)$ time to compute 1) each line L passing through v not strictly intersecting any segment in S; 2) how many segments lie on each side of L. This immediately leads to $O(n^2)$ solutions to Problems 4,5, provided that the set of $L(v)$'s can be computed in the same amount of time. To see this, transform the $2n$ endpoints in S into dual lines. A point $p : (a, b)$ is mapped to the line $D_p : Y = aX + b$, and similarly any line $L : Y = kX + d$ is mapped to the point $D_L : (-k, d)$. Observe that two points are mapped to two parallel lines if and only if these points have the same x-coordinates. We can assume that no endpoints in S share the same x-coordinates, which may possibly entail rotating the axes by a small angle, at a cost of $O(n \log n)$ time. The dual set of the $2n$ endpoints forms a line arrangement, whose underlying planar graph can be computed in $O(n^2)$ time [CGL,EOS]. The key observation now is that for each endpoint v, the line D_v intersects exactly $2n - 1$ lines, which can be retrieved in their intersecting order w.r.t. D_v in $O(n)$ time. But the x-coordinate of $D_v \bigcap D_w$ is precisely $-\text{slope}(vw)$, therefore $L(v)$ can be computed in $O(n)$ time, after $O(n^2)$ preprocessing. This completes the proof. ∎

3. Computing Algebraic Predicates

Given n lines and n points in the plane, does any line pass through any of the points (problem posed by Hopcroft)? Given n points in E^3, are any 5 co-spherical or do any 17 have their interdistances adding up to 1? More generally, given n geometric objects in arbitrary dimensions, do any k interact in a pre-specified manner? This general class of questions can be easily formalized. Let $S = \{V_1, \ldots, V_k\}$ be a collection of k sets of the form $V_i = \{v_1^{(i)}, \ldots, v_{p_i}^{(i)}\}$. Each $v_j^{(i)}$ is a vector $\in \Re^{c_i}$ of the form $v_j^{(i)} = (x_{j,1}^{(i)} \ldots, x_{j,c_i}^{(i)})$. S is the *input* and the quantity $n = \sum_{1 \le i \le k} p_i c_i$ is the *input size*. Let $Q_d(y_{1,1}, \ldots, y_{1,c_1}, y_{2,1}, \ldots, y_{2,c_2}, \ldots, y_{k,1}, \ldots, y_{k,c_k})$ be a rational λ−variate polynomial of degree d; $\lambda = \sum_{1 \le i \le k} c_i$. If $w_i = (y_{i,1}, \ldots, y_{i,c_i})$, we use the abbreviation $Q_d(w_1, \ldots, w_k)$. In the following, the quantities $d, k, c_1, \ldots, c_k$ are considered to be constants.

Problem P: Does there exist a k-tuple $(j_1, \ldots, j_k) \in \Pi_{1 \leq i \leq k}\{1, \ldots, p_i\}$ s.t. $Q_d(v_{j_1}^{(1)}, \ldots, v_{j_k}^{(k)}) = 0$?

To begin with, using elementary elimination theory, one can easily rephrase the problems previously mentioned as special cases of Problem P. There is a trivial $O(n^k)$ solution involving the testing of all possible k-tuples. The following result asserts that it is always possible to do (a little) better.

Theorem 4. Let $c = \min(c_1, \ldots, c_k)$. Problem P can be solved in $O(n^{k - \frac{1}{2^{c+7}}})$ time.

Proof: We generalize the batching strategy introduced by Yao in his work on higher dimensional MST [Y]. Wlog assume that $c = c_k$. Consider the m c_k-variate polynomials of the form $Q_d(v_{j_1}^{(1)}, \ldots, v_{j_{k-1}}^{(k-1)}, z_1, \ldots, z_{c_k})$, where $z_1, \ldots, z_{c_k}$ are the variables and $(j_1, \ldots, j_{k-1}) \in \Pi_{1 \leq i \leq k-1}\{1, \ldots, p_i\}$; note that $m \leq n^{k-1}$. Divide up this set of polynomials into subsets of size α, denoted $W_1, \ldots, W_t$ $(t < 1 + \frac{1}{\alpha}\Pi_{1 \leq i \leq k-1} p_i)$. In [C2] a method is described for preprocessing a set of polynomials so that determining whether a query vector is a zero of one of them can be done in logarithmic time. More precisely, let $F = \{P_1, \ldots, P_\alpha\}$ be a family of α fixed-degree r-variate polynomials with rational coefficients and let $S = \{x \in E^r \mid \Pi_{1 \leq i \leq \alpha} P_i(x) \neq 0\}$. In $O(\alpha^{2^{r+6}})$ time and space, it is possible to compute a set of algebraic points, one in each connected region of S, as well as set up a data structure for computing the predicate $[\exists i \ (1 \leq i \leq \alpha) \mid P_i(q) = 0]$, for any $q \in E^r$. In $O(2^r \log \alpha)$ time, the algorithm will return an index i such that $P_i(q) = 0$ if such an index is to be found, otherwise it will return the algebraic point associated with the unique region of S that contains q. Applying this method to each set $W_1, \ldots, W_t$ requires $O(t\alpha^{2^{c+6}}) = O(n^{k-1}\alpha^{2^{c+6}-1})$ preprocessing time. On the other hand, this will allow us to test the p_k vectors of V_k in $O(p_k t \log n) = O(\frac{n^k \log n}{\alpha})$ overall query time. Setting $\alpha = (n \log n)^{1/2^{c+6}}$ leads to a time complexity of $O(n^{k - \frac{1}{2^{c+6}}}(\log n)^{1 - \frac{1}{2^{c+6}}})$, which for the sake of simplicity we conservatively estimate as $O(n^{k - \frac{1}{2^{c+7}}})$ time. ∎

4. Conclusions

This work attempts to place the notion of *criticality* in implicit set representations at a more central location. By doing so we are able to improve on a number of previous complexity results. For the most part, our improvements are algorithmic in nature; we feel that more geometric insights will be needed to decide whether the criticality of point sets and their duals, line

arrangements, constitutes an unsurmountable obstacle to computational efficiency. For example, is quadratic space required to answer the point-on-query-line question in polylogarithmic time?

REFERENCES

[C1] Chazelle, B. *Filtering search: A new approach to query-answering*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 122–132.

[C2] Chazelle, B. *Fast searching in a real algebraic manifold with applications to geometric complexity*, Brown Tech. Rep. CS–84–13, June 1984.

[CG] Chazelle, B., Guibas, L.J. *Fractional cascading: A data structuring technique with geometric applications*, in preparation.

[CGL] Chazelle, B., Guibas, L.J., Lee, D.T. *The power of duality*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 217–225.

[CY] Cole, R., Yap, C.K. *Geometric retrieval problems*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 112–121.

[EGS] Edelsbrunner, H., Guibas, L., Stolfi, J. *Optimal point location in a monotone subdivision*, to appear in SIAM J. Comp.

[EKM] Edelsbrunner, H., Kirkpatrick, D.G., Maurer, H.A. *Polygonal intersection search*, Inform. Process. Lett. 14 (1982), pp. 74–79.

[EOS] Edelsbrunner, H., O'Rourke, J., Seidel, R. *Constructing arrangements of lines and hyperplanes with applications*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 83–91.

[EOW] Edelsbrunner, H., Overmars, M.H., Wood, D. *Graphics in Flatland: a case study*, Adv. in Comput. Res., Vol. 1, ed. F.P. Preparata, JAI Press, (1983), pp. 35–59.

[EW] Edelsbrunner, H., Welzl, E. *Halfplanar range search in linear space and $O(n^{0.695})$ query time*, Rep. F111, Inst. Inform. Proc., Tech. Univ. Graz, Austria, 1983.

[GS] Guibas, L.J., Stolfi, J. *Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams*, Proc. 15th Ann. SIGACT Symp. (1983), pp. 221–234.

[MP] Muller, D.E., Preparata, F.P. *Finding the intersection of two convex polyhedra*, Theoret. Comput. Sci., 7 (1978), pp. 217–236.

[W] Willard, D.E. *Polygon retrieval*, SIAM J. Comp., 11 (1982), pp. 149–165.

[Y] Yao, A.C. *On constructing minimum spanning tree in k-dimensional space and related problems*, SIAM J. Comput. 11 (4), 1982, pp. 721–736.

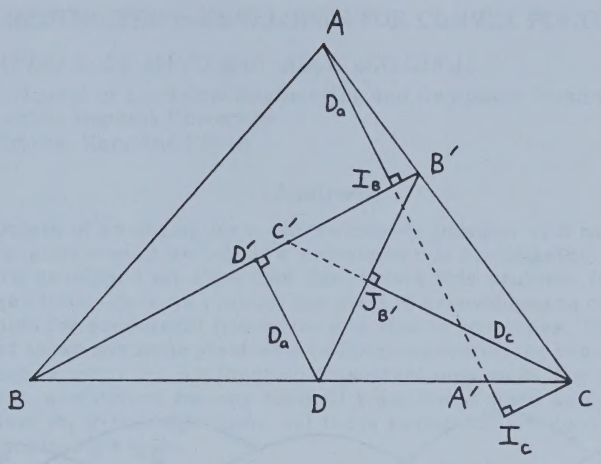


Figure 1

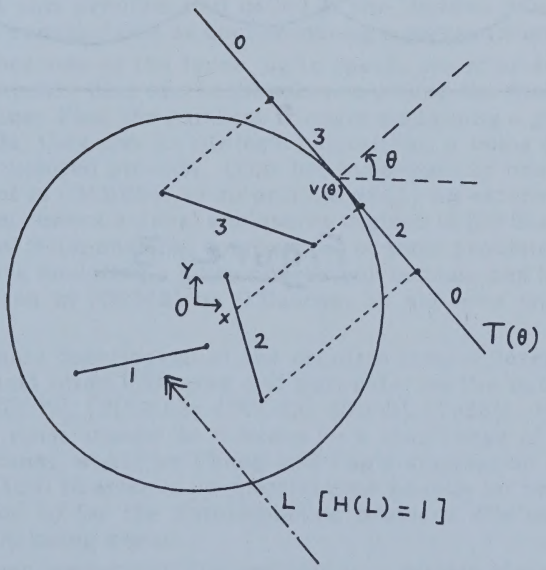


Figure 2

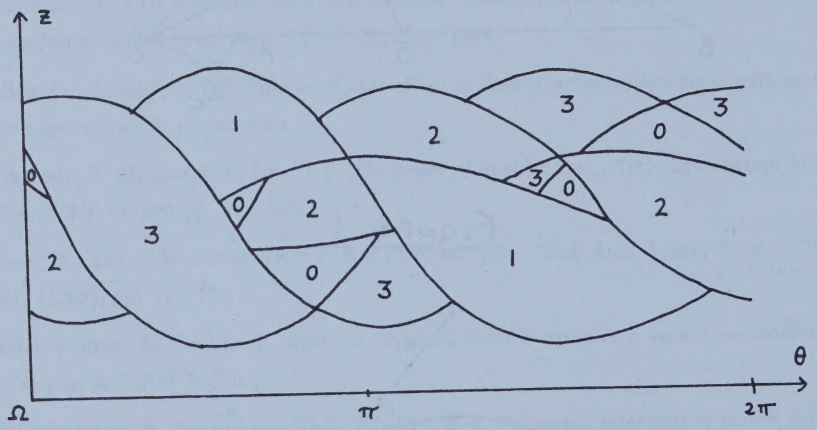


Figure 3

FINDING RESTRICTED k-ENVELOPES FOR CONVEX POLYGONS

N. ADLAI A. DE PANO and *ALOK AGGARWAL*
Department of Electrical Engineering and Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

The problem of enclosing an n-vertex convex polygon with minimal area convex k-gons -- what we call as k-envelopes -- is investigated. Chang and Yap have developed an algorithm that solves this problem for any k in $O(n^3 \log k)$ time. Here we restrict the class of k-envelopes to obtain faster algorithms for equilateral triangular and square envelopes. The generalization of these methods yield an $O(nT(k))$ algorithm for regular k-gonal envelopes, where $T(k)$ is a function dependent only on k. For equiangular envelopes, and indeed for any class of k-gons with fixed angle sequence, we achieve an $O(kn)$ algorithm. All these results extend to minimal perimeter enclosing k-gons.

1. Introduction

A number of papers have recently appeared dealing with the problem of optimizing polygonal approximations of various classes of geometric objects. The classic example is the "potato-peeling" problem defined by Goodman in [Go81], where he sought to find the largest inscribed convex polygon inside an arbitrary simple polygon. Woo independently looked at this problem and called it the "convex skull" problem [Wo83]. One can consider this as approximating a polygon from within.

On the other side of the fence, so to speak, are efforts in approximations from without. Klee and Laskowski are among the first to investigate along this line: Find the smallest triangle containing a given convex polygon. In [KL84] they give an $O(n \log^2 n)$ algorithm, n being the number of edges of the enclosed polygon. (This has subsequently been improved by O'Rourke *et al* in [OAMB84] to an optimal $\Theta(n)$.) An extension of their results to general convex k-gonal enclosures is given in [DP84a]. Furthermore, the natural 3-dimensional counterpart of their problem, minimum volume tetrahedra enclosing a given convex polyhedron, has been the topic of investigation in [OR84b] by O'Rourke; he achieves an $O(n^4)$ procedure.

Other authors have looked at the problem from different perspectives, working most often with area and perimeter as the optimality criterion ([FS75], [DS79], [BDDG82], [DDG83], [DB83], [To83]). Perhaps the most significant work, insofar as it bears on a wide range of variants of the original problem, would be Chang and Yap's application of dynamic programming [CY84] to achieve polynomial time bounds for various problems (e.g. $O(n^9 \log n)$ for the potato-peeling problem; $O(n^3 \log k)$ for the minimum area enclosing k-gon).

In this paper, we restrict the problem to a certain class of polygon approximations, with the intent of obtaining fast algorithms in exchange for the added structure. The restriction is a reasonable one and finds application in the "stock-cutting" problem. (This problem has an extensive bibliography. See for example [DB83] and references therein.) For brevity, we will sketch some proofs, leaving the full demonstrations for a technical report that is in preparation.

2. Regular k-Envelopes

We now define k-envelopes as a way of shortening notation.

Definition 1: The *k-envelope* for a given geometric object is the minimum area k-sided convex polygon that contains it.

Henceforth, instead of saying "the minimum area convex k-gon containing X", we simply write "the *k-envelope* for X".

The restriction is to find the regular k-envelope of a given convex polygon. When $k=3$, we are looking for the smallest equilateral triangle. When $k=4$, we are dealing with the smallest square; $k=6$, the smallest regular hexagon. Note that for these values of k, the k-envelope can tile the plane [DL81].

We can show that even for the regular 3-gon, different geometric characterizations come into play compared to those for general triangles as given in [KL84] and [CY84]. For example, the regular 3-envelope may have edges that are *not* balanced (that is, touch the enclosed polygon with their midpoints). Figure 1 illustrates this. Moreover, the regular k-envelope may not even share any edges with the enclosed polygon (see Figure 2 where $k=4$).

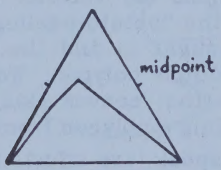


Figure 1: Some edges of the regular k-gon may not be balanced.

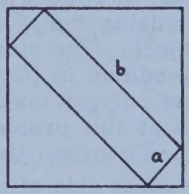


Figure 2: The regular k-envelope may not have shared edges with the n-gon.

Henceforth, n will always be used to indicated the number of edges in the enclosed polygon and k, the number of edges in the envelope.

3. Summary of Results

We give here a listing of our main contributions. Due to space limitations, we give only brief descriptions of algorithms. Details will be given elsewhere.

3.1. Regular 3-Envelopes

We give a characterization that leads to an optimal $\Theta(n)$ algorithm which computes the regular 3-envelope for a convex n-gon. A variation of Toussaint's rotating caliper algorithm is employed (see [To83]).

3.2. Equiangular and P_θ k-Envelopes

The characterization for regular 3-envelopes can be extended to equiangular k-envelopes and, indeed, to k-envelopes whose internal angles form a fixed sequence $\Theta=(\vartheta_1,\ldots,\vartheta_k)$, for all $i,\vartheta_i<\pi$ and $\sum_1^k \vartheta_i=(k-2)\pi$. In [CY84] this class of k-envelopes is labelled P_θ. An $O(kn)$ procedure is described.

3.3. Regular 4-Envelopes

We describe how to determine the regular 4-envelope (square) via a procedure that is based on a geometric characterization of the regular 4-envelope relative to equiangular 4-envelopes. An $O(n)$ algorithm is given.

3.4. Regular k-Envelopes, $k>4$

The method of analysis for higher values of k is a natural extension of the one employed for $k=4$. This gives rise to a procedure that runs in time proportional to $n \times T(k)$, where $T(k)$ depends only on k.

3.5. Minimum-Perimeter Regular k-Gons

The restriction to regular k-envelopes has an added bonus in that the regular k-envelope also gives the minimum perimeter regular k-gonal enclosure.

4. The Regular 3-Envelope

We now characterize the regular 3-envelope for a given convex n-gon.

Lemma 1: Let P_1, P_2, P_3 be 3 points in E^2 and let E_1, E_2, and E_3 be lines passing through them respectively so as to define an equilateral triangle (see Figure 3). Consider the area of this triangle as we rotate each of the 3 lines about their respective points by some angle ϑ. Then if this area increases as we rotate in the clockwise (resp. counterclockwise) direction, it must decrease if we rotate in the counterclockwise (resp. clockwise) direction.

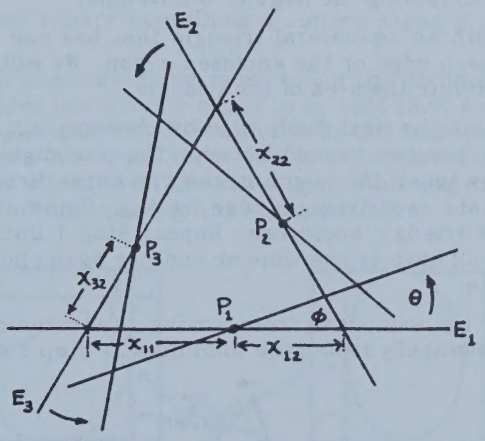


Figure 3: The proper rotation will decrease the area of the triangle defined by the lines.

Proof: Consider the change in area induced by a clockwise rotation of ϑ. After considerable simplification this can be expressed as a function $\alpha(\vartheta)$;

$$\alpha(\vartheta) = \sin\vartheta\ [\ A\sin(\vartheta+\varphi)+B\sin(\vartheta-\varphi)-2C\sin\vartheta\ ]/2\sin\varphi$$

where $\varphi = 60°$; $A = \sum_1^3 x_{i,1}^2$; $B = \sum_1^3 x_{i,2}^2$; and $C = \sum_1^3 x_{i,1}x_{i-1,2}$. If a rotation of ϑ is

made counterclockwise, it can be shown that the change is simply $\alpha(-\vartheta)$. We will show that when ϑ is small, either $\alpha(\vartheta)$ or $\alpha(-\vartheta)$ will be negative, and never will they simultaneously be positive (i.e., there is always a direction that will decrease the area). We will do this by taking $\alpha'(\vartheta)$ and establishing that if this derivative is zero at $\vartheta=0$ then $\alpha(\vartheta)$ is a local maximum there. Thus, since $\alpha(0)=0$, if rotating by $d\vartheta$ increases area, then rotating the other way (by $-d\vartheta$) should decrease it. Differentiating $\alpha(\vartheta)$ yields:

$$\alpha'(\vartheta) = \cos\vartheta\,[\,A\sin(\vartheta+\varphi)+B\sin(\vartheta-\varphi)-2C\sin\vartheta\,]/2\sin\varphi + \sin\vartheta\,[\,A\cos(\vartheta+\varphi)+B\cos(\vartheta-\varphi)-2C\cos\vartheta\,]/2\sin\varphi$$

At $\vartheta=0$, we have $\alpha'(0) = (A-B)/2$. If $A\neq B$ our result is proved. If $A=B$, since $\alpha''(0) = -(A+B)/2$, which is negative, we have a local maximum at $\vartheta=0$, and any rotation will decrease the area. ▪

Think of the 3 points as places where a candidate 3-envelope touches the enclosed n-gon. Any rotation must be prevented in the proper direction as any such "perturbation" will decrease area, contradicting the assumed minimality of the 3-envelope. The following lemma is immediate:

Lemma 2: The 3-envelope shares at least one edge with the enclosed convex n-gon.

Proof: The only way rotations can be prevented is when at least one edge of the 3-envelope is flush with the enclosed n-gon. ▪

We now outline an algorithm that determines the 3-envelope by a procedure similar to Toussaint's rotating caliper algorithm [To83]. A simple analysis will show that this procedure works in $O(n)$ time.

Algorithm 1: (Computing the Regular 3-Envelope)

Step 0: Start with an equilateral triangle that has one edge flush with a chosen edge or the enclosed n-gon. We will rotate clockwise. Compute the area of the triangle.

Step 1: Determine the next flush edge by choosing ϑ_{min}, the minimum of the 3 angles formed between the 3 triangle edges and where they touch the n-gon in the clockwise direction (see Fig. 4). Rotate each triangle edge by ϑ_{min}. Compute the area of this new triangle enclosure. Repeat Step 1 until all edges of the n-gon have at one time or another been chosen as the shared edge.

Step 2: Choose the minimum from among $O(n)$ candidate triangles (or dynamically retain the minimum as Step 1 is repeated).

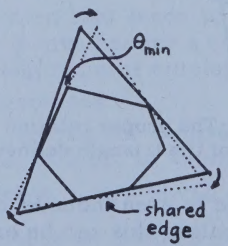


Figure 4: The triangle configuration in Step 1 of Algorithm 1 showing a shared edge and ϑ_{min}.

We have thus shown the following result:

Theorem 1: The 3-envelope for a convex n-gon can be determined in $O(n)$ time. Furthermore, this result is optimal to within a multiplicative constant.

4.1. Equiangular and P_θ k-Envelopes

Lemma 1 can be extended to the more general case of equiangular k-envelopes, where all internal angles are equal to $(k-2)\pi/k$, and, indeed, to the class P_θ, as defined in [CY84] (see Sec. 3.2). A formal extension of Lemma 1 will show that these types of k-envelopes share at least one edge with their enclosed convex n-gon. (Now, A, B, and C in the proof of Lemma 1 will have limits going up to k, and for the class P_θ, $\sin\varphi$ goes into the summations as $\sin\varphi_i$.) The immediate consequence of this is that a rotating caliper algorithm can be designed which functions in $O(nk)$ time to compute both the equiangular k-envelope and the P_θ k-envelope. The factor k enters the analysis because for every caliper step, we have to choose a minimum from among k angles and compute the k-gon's area.

We have, in effect, shown the following:

Theorem 2: The equiangular k-envelope for a convex n-gon can be determined in $O(nk)$ time.

Theorem 3: The P_θ k-envelope for a convex n-gon can be determined in $O(nk)$ time.

It remains unclear, however, whether these results are optimal or not.

5. The Regular 4-Envelope

The smallest square containing a convex n-gon is characterized in this section.

First of all, the 4-envelope must touch the enclosed n-gon with at least 3 of its 4 edges (we assume edges to include their endpoints). If not, i.e., if there are 2 edges *not* in contact with the enclosed n-gon, we can move these 2 edges "towards" the n-gon so as to decrease the area of the resulting square enclosure, at the same time achieving an additional contact point. Figure 5 summarizes the argument here.

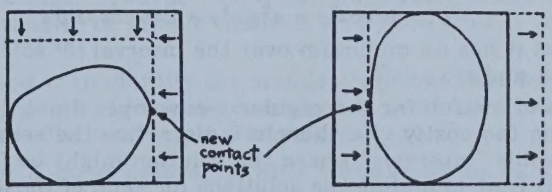


Figure 5: The 4-envelope touches the enclosed convex polygons with at least 3 of its 4 edges.

However, we cannot hope to find, in general, a shared edge the way we did for the regular 3-envelope as there are cases where only isolated contact points are shared. Figure 2 gives a simple example. We should

specify, however, that the rectangle there has to have the ratio of the longer side b to the shorter side a greater than $1+\sqrt{2}$. The following remark summarizes this result:

Remark: Let R be a rectangle with dimensions b and a, with $b=ca$ and $c>1+\sqrt{2}$. Then the smallest square containing R touches R at its 4 vertices and is tilted at an angle of 45°.

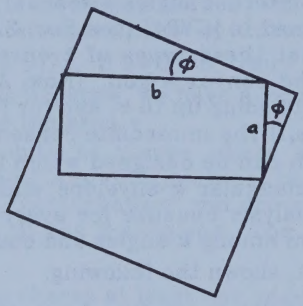


Figure 6: A square enclosure for a rectangle that is "tilted" at an angle φ.

Proof: The regular 4-envelope for a rectangle either shares at least 2 edges with it or none at all. If at least 2 edges are shared, the area of the enclosure will be equal to b^2. If the enclosure is tilted at an angle φ, which is between 0° and 45°, the area will be:

$$(b\cos\varphi + a\sin\varphi)^2 = b^2\cos^2\varphi + a^2\sin^2\varphi + 2ab\cos\varphi\sin\varphi$$

For this area to be smaller than b^2, we must have:

$$b^2 - (b^2\cos^2\varphi + a^2\sin^2\varphi + 2ab\cos\varphi\sin\varphi) > 0$$
$$(b^2 - a^2)\sin^2\varphi - 2ab\cos\varphi\sin\varphi > 0$$

With further simplification, it becomes clear that we must have φ such that $\tan\varphi < 2ab/(b^2 - a^2)$. Since φ must be less than 45°, we must have $2ab/(b^2 - a^2) < 1 = \tan 45°$. Now, if we let $b=ca$, where $c > 1$, we have $2c/(c^2 - 1) < 1$, which when solved for the proper value of c yields $c > 1 + \sqrt{2}$. Upon analysis of the area function of the tilted enclosure

$$A(\varphi) = b^2\cos^2\varphi + a^2\sin^2\varphi + 2ab\cos\varphi\sin\varphi$$

one finds that it has its minimum over the interval $[0°,45°]$ at $\varphi = 45°$, that is, when $b=ca$ and $c > 1+\sqrt{2}$. ▪

Thus, our search for the regular 4-envelope, if not done intelligently, risks being too costly. We therefore discretize the search by identifying $O(n)$ possible "intervals" where the solution might be found, and then spending $O(1)$ time in finding the solutions for each of those intervals.

Definition 2: An *interval* in Toussaint's caliper tour (which computes the smallest rectangle containing a convex polygon) is defined to be the continuum of rectangle orientations between successive steps of the tour.

Figure 7 illustrates one Toussaint interval. We choose to refrain from giving a formal definition using metrics, trusting that this descriptive definition is unambiguous enough for our purposes.

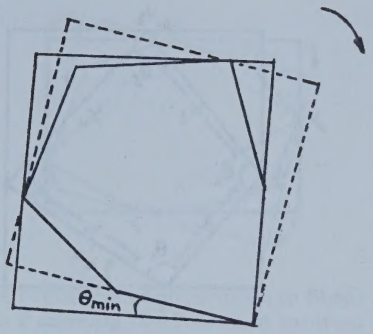


Figure 7: A Toussaint interval.

The following lemma is the key to our search:

Lemma 3: There is a rectangle configuration in at least one Toussaint interval which when extended gives a regular 4-envelope for a convex n-gon.

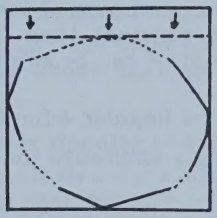


Figure 8: **The regular 4-envelope can always be "reduced" to a rectangle that lies in some Toussaint interval.**

Proof (Outline): We work backwards. Let S be the regular 4-envelope for a given n-gon. At most, there is one edge of S which does not touch the n-gon. Move this edge "towards" the n-gon until there is contact. This is the desired rectangle. A simple analysis will show that it lies in some Toussaint interval. ▪

The task now is to identify the smallest squares that are extensions (in the sense of merely moving out one edge) of rectangles in a particular Toussaint interval The next lemma takes care of this:

Lemma 4: Given a quadrilateral Q with vertices v_1, v_2, v_3, v_4, edges of lengths x_1, x_2, x_3, x_4, and interior angles $\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4$ (as shown in Figure 9), then the smallest square containing Q which touches v_1 is tilted at an angle α^*, where α^* is given by:

$$\alpha^* = \tan^{-1}\left[\frac{-x_1+x_2(\cos\vartheta_1+\sin\vartheta_1)-x_3\sin(\vartheta_1+\vartheta_2)}{x_2(\sin\vartheta_1-\cos\vartheta_1)+x_3\cos(\vartheta_1+\vartheta_2)}\right]$$

Proof (Outline): The smallest square is obtained when there are 4 contact points. This happens when the tilt is precisely α^*. The full proof is too technical to include here. ▪

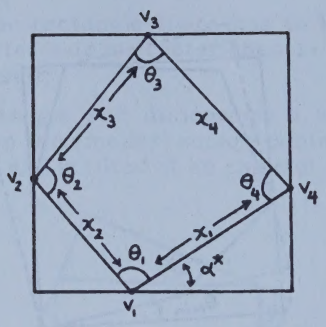


Figure 9: Looking for the smallest square containing a quadrilateral.

Our search will thus be conducted via a caliper tour where we solve a problem similar to that in Lemma 4 at each caliper stop. The quadrilateral involved will be determined by the 4 contact points of the rectangle with the enclosed n-gon in that particular Toussaint interval. The relative lengths of the rectangle's sides will tell us how to identify the parameters needed to apply Lemma 4. (Even if we naively applied Lemma 4 on all 4 contact points, and took the minimum of the possibly 4 squares obtained, we still would have spent constant time for this step.) Since the trigonometric formula in Lemma 4 can be solved in $O(1)$ time, we achieve an $O(n)$ procedure.

Algorithm 2: (Computing the Regular 4-Envelope)

Step 0: Start with a rectangle enclosure that shares an edge with the enclosed n-gon.

Step 1: Determine the next caliper step by computing ϑ_{min}, the minimum of the 4 angles formed at the 4 "contact points" in the direction of rotation. Rotate each edge of the rectangle by this amount to give the next "rotating caliper" configuration.

Step 2: In the Toussaint interval determined in Step 1, the rectangle maintains contact with the n-gon at exactly 4 points. Identify v_1 of Lemma 4 with one of the contact points along the longer edges of the enclosure. Identify the rest of the parameters accordingly. Solve for α^* and check if the square induced by α^* "reduces" to a rectangle in the current Toussaint interval (a simple comparison betwen α^*, ϑ_{min}, and one other parameter). If not, one of the interval "endpoint" configurations gives rise to that interval's minimum square (by simple extension of one edge).

Step 3: Repeat Step 1 and Step 2 until the caliper tour is completed. Choose from among the $O(n)$ candidate enclosures (or dynamically retain the current minimum).

We have thus established the following:

Theorem 4: The regular 4-envelope for a convex n-gon can be determined in $O(n)$ time. Furthermore, this result is optimal to within a multiplicative constant.

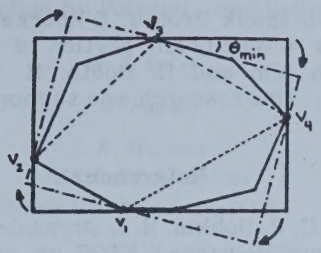


Figure 10: A rectangle configuration in Steps 1 and 2 of Algorithm 2 showing the Toussaint interval and the quadrilateral on which Lemma 4 would be applied.

5.1. Regular k-Envelopes, $k>4$

Note that we obtained the regular 4-envelope algorithm through a caliper tour which obtained the smallest rectangle, or equiangular 4-envelope. This can be extended to regular k-envelopes, $k>4$, where now the basis of the search will be to determine minimum regular k-envelopes that enclose given k-gons (such as, for example, the k-gon that is determined by the points of contact of an equiangular k-gon with its enclosed n-gon). At each step, a more complex computation is required to obtain such a minimum regular k-gon. The time complexity is dependent only on k, however, and it can be shown that an $O(n \times T(k))$ solution can be achieved.

6. Minimum Perimeter Regular and Equiangular k-Gons

We have injected so much structure into the class of polygons with which we are approximating that for this case, minimum area implies minimum perimeter as well. Recall that both area and perimeter of a regular polygon are directly proportional to its diameter. Clearly, then, the regular k-envelope, which minimizes the area, also minimizes perimeter. For equiangular k-envelopes, the relationship is more subtle but can still be worked out. We omit the details here.

7. Final Remarks and Acknowledgements

Fast algorithms can be achieved for polygon approximation problems when we restrict ourselves to structured k-envelopes. It must be said here that even though there are practical applications for our results, the problem in itself is quite interesting if only for its conceptual simplicity. The geometric characterizations, nevertheless, can become quite subtle, and it would be extremely satisfying to see how all these concepts translate into higher dimensions, e.g., regular polyhedral enclosures in E^3.

Some work has also been done using some other restrictions (e.g., non-regular envelopes that admit tilings of the plane) to achieve fast algorithms [DP84b]. It is fascinating to note that although there are procedures that compute the k-envelope in $O(n)$ time for k=3,n, and n-1, no sub-quadratic algorithms have been formulated for the intermediate values of k (the technique described in [DB83] is faulty, as pointed out by Chang and Yap [CY84], and substantiated by O'Rourke [OR84a]).

We would like to thank Prof. J. O'Rourke for stimulating and encouraging discussions, a significant portion of which took place during weekly meetings with him and H. Booth, M. McKenna, C. Phillips, R. St.Amant, and S. Suri. This research was supported in part by NSF Grant MCS 83-04780.

References

[BDDG82] J. E. Boyce, D. P. Dobkin, R. L. Drysdale, and L. J. Guibas. "Finding Extremal Polygons," *STOC*, pp. 282-289 (1982).

[CY84] J. S. Chang and C. K. Yap. "A Polynomial Solution for Potato-Peeling and Other Polygon Inclusion and Enclosure Problems," to be presented at *FOCS*, (1984).

[DB83] D. Dori and M. Ben-Bassat. "Circumscribing a Convex Polygon by a Polygon of Fewer Sides With Minimal Area Addition," *Computer Vision, Graphics and Image Processing*, **24**, pp. 131-159 (1983).

[DDG83] D. P. Dobkin, R. L. Drysdale, and L. J. Guibas. "Finding Smallest Polygons," *Advances in Computing Research*, F. P. Preparata (Ed.), Jai Press, (1983).

[DL81] I. Debroey and F. Landuyt. "Equitransitive Edge-to-Edge Tilings by Regular Polygons," *Geometriae Dedicata*, **11**, pp. 47-60 (1981).

[DP84a] N. A. A. De Pano. "Approximations of Polygons and Polyhedra: Potentials for Research," *Manuscript*, (1984).

[DP84b] N. A. A. De Pano. "Efficient Polygon Enclosures That Cover the Plane," The Johns Hopkins University, Technical Report, in preparation (1984).

[DS79] D. P. Dobkin and L. Snyder. "On a General Method for Maximizing and Minimizing Among Certain Geometric Problems," *FOCS*, pp. 7-19 (1979).

[FS75] H. Freeman and R. Shapira. "Determining the Minimum Area Encasing Rectangle for an Arbitrary Closed Curve," *CACM*, **18**, pp. 409-413 (July, 1975).

[Go81] J. E. Goodman. "On the Largest Convex Polygon Contained in a Non-Convex n-Gon or How to Peel a Potato," *Geometriae Dedicata*, **11**, pp. 99-106 (1981).

[KL84] V. Klee and M. C. Laskowski. "Finding the Smallest Triangles Containing a Given Convex Polygon," *J. of Algorithms*, to appear (1984).

[OAMB84] J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin. "An Optimal Algorithm for Finding Minimal Enclosing Triangles," The Johns Hopkins University, Technical Report (1984).

[OR84a] J. O'Rourke. "Counterexamples to a Minimal Circumscription Algorithm," *Manuscript*, (1984).

[OR84b] J. O'Rourke. "Finding Minimal Enclosing Tetrahedra," The Johns Hopkins University, Technical Report (1984).

[To83] G. Toussaint. "Solving Geometric Problems with Rotating Calipers," Proc. IEEE MELECON '83, (May 1983).

[Wo83] T. C. Woo. "Convex Skulls," *Manuscript*, (April 1983).

Inspection by Polygon Containment

B. S. Baker

S. J. Fortune

S. R. Mahaney
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. Introduction

We investigate the problem of whether one polygon P can be translated to fit inside another polygon P'. We obtain algorithms that run in $O(n^2 \log n)$ time in the case where P is convex and in the case where both polygons are rectilinearly convex. The parameter n is the sum of the number of bounding edges in the two polygons. Our algorithms compute the boundary of the feasible region, that is, the set of positions in which P fits inside P'. Both algorithms are nearly optimal in the sense that the feasible region may have $\Omega(n^2)$ edges.

We use the first algorithm to solve the following *inspection* problem in two dimensions. A manufactured part is desired to have a convex polygonal shape S. Tolerances are given in terms of two convex polygons, one lying within S and the other lying outside S, as shown in Figure 1. A vision system yields a description of each manufactured part as a polygon P, not necessarily convex because of noise and fabrication error. It is desired to determine whether P can be translated to lie between the tolerance polygons, in which case the part is considered to conform to the desired tolerances. Our strategy is to divide this problem into two subproblems: first, whether the inner tolerance polygon fits inside P (a convex polygon inside a nonconvex polygon), and second, whether the convex hull of P fits in the outer tolerance polygon (a convex polygon inside a convex polygon). We compute the feasible region for each problem and then determine whether the intersection of the two feasible regions is empty.

The algorithm in the case that P is convex is easily extended to the case in which P' has holes, and P must fit inside P' but outside all of the holes. This extended algorithm has applications to a stock-cutting problem and to a collision-avoidance problem in robotics.

Suppose a manufacturer needs to cut successive convex objects out of a sheet of material. If the material has a grain, the orientation of the objects may be fixed. After cutting out the first object, the remaining part of the sheet may no longer be convex, and indeed may have holes. Our algorithm solves the problem of whether a new convex object can be cut out of the remaining sheet.

The problem of finding a collision-free path for a polygonal object among polygonal obstacles arises in robotics. This problem is discussed by Lozano-Perez and Wesley [9]; they consider both the case in which the polygon may rotate and the case in which it may not. For a fixed orientation of the object, their general approach is to "grow" each obstacle to represent how close a reference point on the polygon can come to the obstacle from each direction without a collision. The region not covered by any "grown" obstacle represents the region in which the reference point can move without a collision between the object and an obstacle. They do not suggest how to compute the boundaries of this feasible region efficiently. Our algorithm computes the boundary of the feasible region

when the polygon object must move in a fixed orientation.

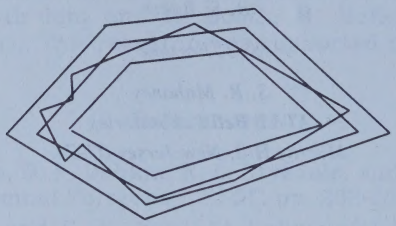


Figure 1.

Related results on the polygon containment problem have been obtained by Chazelle [2]. He gives a linear time algorithm in the case where both polygons are convex. He also investigates polygon containment where both translation and rotation are allowed; his $O(n^3)$ algorithm for the case in which both polygons are convex suggests that rotation makes the polygon containment problem much more difficult.

We also consider briefly the problem of determining whether one nonconvex polygon can be translated to fit within another. We show that examples exist for which the feasible region has $\Omega(n^4)$ edges, where n is the number of edges in each polygon, and both polygons are rectilinear.

2. Preliminaries

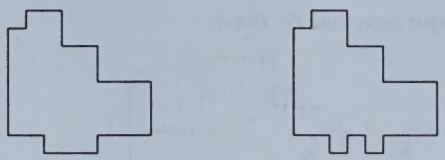
Suppose C and D are polygons, D is fixed in the plane, and C is free to move. We wish to find all placements of C so that C is contained in D. We consider only the case of placements of C obtained by translation of C from some fixed orientation of C, without rotation.

Notice that a placement of C is uniquely determined by a placement of a specific point of C. We call this specific point the *reference point*. Hence whenever we speak of a placement of C, or a set of placements of C, we mean the placement or set of placements of the reference point. A *feasible placement of C inside D* is a placement of C so that C is contained in D. The set of feasible placements of C inside D by $I(C,D)$. Similarly, a *feasible placement of C outside D* is a placement of C so that no interior point of C intersects D. The set of feasible placements of C outside D is denoted $O(C,D)$.

A polygon is *rectilinear* if all bounding edges are vertical or horizontal. It is *rectilinearly convex* if horizontal or vertical segments joining two segments of the polygon are contained in the polygon. See Figure 2. Note that a polygon can be rectilinearly convex without being convex.

3. The Rectilinearly Convex Case

Suppose C and D are simple rectilinearly convex polygons, with c and d edges, respectively. How many edges can the feasible region $I(C,D)$ have? Consider placing C inside D, and then pushing C to the right until it touches D. In this position, a vertical edge c_i of C touches a vertical edge d_j of D, and C is positioned at a vertical segment of the boundary of the feasible region. The length of the vertical segment is determined by how far C can slide upward or downward without c_i losing contact with d_j or a horizontal edge of C colliding with a horizontal edge of D. Thus, each vertical edge in the boundary of the feasible region is derived from a contact between a vertical edge of C and a vertical edge of D. Similarly, each horizontal edge in the boundary of the feasible region is derived from a horizontal edge of C and a horizontal edge of D. Consequently, the boundary of $I(C,D)$ has $O(cd)$ edges.



a b

Figure 2. Rectilinear polygons. (a) is rectilinearly convex, (b) is not.

Moreover, there are examples for which the boundary of $I(C,D)$ region has $\Omega(cd)$ edges. Consider polygons C and D as shown in Figure 2, each with $c=d=2k+2$ edges. Each has "steps" on the lower right boundary, but the "steps" of D are k times as long as those of C. Obviously, each "step" of C can touch a "step" of D at the boundary of the feasible region, implying that the boundary of the feasible region has at least $k^2=\Omega(cd)$ edges.

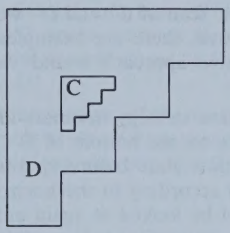


Figure 3. The lower bound example for rectilinear polygons.

Theorem 3.1. Let C and D be simple rectilinearly convex polygons with c and d edges, respectively. Then there is an $O(cd \log cd)$ algorithm to compute $I(C,D)$.

Proof: The algorithm consists of three parts:
1. divide D and C each into 4 quadrant parts;
2. determine the feasible regions for matching quadrant parts, C' and D' (with open edges extended to rays);
3. intersect the four feasible regions of step 2, giving the feasible region of C inside D.

For a rectilinearly convex polygon P, the lower right quadrant, denoted P', which we will use for illustration, consists of the bottommost edge extended to a ray to the left, the "steps" up and to the right, and the rightmost edge extended to a ray upward. (See Figure 4.) This decomposition of C and D can be done in $O(c+d)$ steps assuming the edges are sorted or presented in clockwise order. Otherwise, the edges may be sorted within the time bound of the algorithm.

The second part of the algorithm takes time $O(cd \log cd)$ and is the main part of the proof. This step finds the boundary of the feasible region for C' and D' by sliding C' along the constraining edges of D'. Initially, C' is placed with its bottom touching that of D'. Then C' slides to the right until a vertical edge c_i of C' hits a vertical edge d_j of D'. At this point, C' slides upward until c_i clears d_j, *i.e.* until the bottom of c_i touches the top of d_j. Then C' can slide to the right again. Thus, in each step, either C' slides to the right, with a horizontal edge of C' sliding along a horizontal edge of D', or C' slides upward, with a vertical edge of C' sliding along a vertical edge of D'. Since the edges touch, C' is sliding along the boundary of the feasible region. An edge bounding

the feasible region is output each time C' slides.

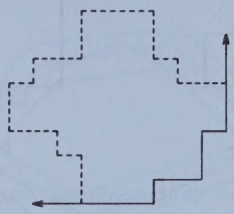


Figure 4. The lower right quadrant of a rectilinearly convex polygon

The difficulty lies in efficiently determining at each height which edge of C' will be the first to collide with an edge of D' when C' slides to the right. A naive approach would be to compute for each edge c_i of C' how far it can slide to the right before colliding with an edge d_j of D'. A linear scan of C' and D' would accomplish this in $O(c+d)$ time. Unfortunately, as shown above, there are examples in which C' must slide to the right $\Omega(cd)$ times. Thus, this naive approach would yield only an $O(cd(c+d))$ time bound.

Instead, we use a lazy evaluation strategy to obtain an $O(cd \log cd)$ time bound. Initially, when the bottom of C' rests on the bottom of D', a calculation is made for each edge c_i of C' as to how far C' must slide before c_i hits D'. For each c_i, an entry is inserted in a priority queue PQ [1] according to the horizontal coordinate at which c_i hits the edge of D'. This entry will not be looked at again until it becomes the minimum element of PQ. In the meantime, whenever C' needs to slide to the right, the minimum element is extracted from PQ (and deleted from PQ). Suppose the minimum element represents a collision of c_k with d_r. If this element is up-to-date, *i.e.* c_k can collide with d_r at this height, then C' slides to the right until c_k collides with d_r, and then up until c_k clears d_r; at this point, a new constraint is calculated for c_k and inserted into PQ. If this element is out-of-date, *i.e.* C' is now too high for c_k to collide with d_r, then a new computation is made of how far C' must go to the right before c_k collides with an edge of D', and a new element representing this information is inserted in the priority queue. Since C' still needs to slide to the right, the minimum element is extracted from PQ again and examined for up-to-dateness as before. This strategy succeeds because the up-to-date value for an edge is always larger than an out-of-date value.

Exactly what information do we need to store in PQ, and how do we determine whether an element of PQ is up-to-date?

If c is a vertical edge of C' or D', we let $c.bot$ denote the bottom vertex of c, and $c.bot.h$ its horizontal coordinate. (If c is a vertical edge of C', the coordinate represents the position of c when C' is placed at (0,0).) We use selectors $.top$ and $.v$ similarly.

An element of PQ will contain a constraint of the form (H, c_i, d_j, V). If this element is inserted into PQ when C' is at height v', it represents that edge c_i of C' will hit edge d_j of D' when C' is moved to the right to position (H, v'). Moreover, c_i will not hit any edge of D' until C' slides that far to the right. As the algorithm progresses, if the current height of C' is v, we say that (H, c_i, d_j, V) is up-to-date as long as $v < V$.

Suppose that C' is located at (h', v') and we wish to compute the constraint for edge c_i. We assume $d_j = \text{touch}(c_i, v')$ is the first edge of D' that c_i would hit if it were translated to the right. Then c_i will hit d_j when C' is moved to $(d_j.top.h - c_i.bot.h, v')$. Later, when the current height of C' is v, the constraint will be up-to-date as long as $v < d_j.top.v - c_i.bot.v$. Thus, $(d_j.top.h - c_i.bot.h, c_i, d_j, d_j.top.v - c_i.bot.v)$ is the appropriate constraint at (h', v'). See Figure 5.

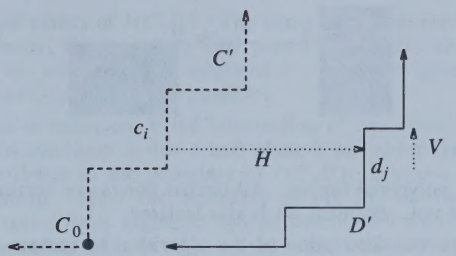


Figure 5. The constraint (H, c_i, d_j, V). The reference point C_0 lies at the origin. C' may be translated right by H at which point c_i hits d_j The constraint is "up-to-date" until C' is translated upward by V.

A simple analysis shows that there are $O(cd)$ operations on the priority queue and $O(cd)$ computations of touch. Extract_min costs $O(1)$ each time. Insertions and deletions cost $O(\log c)$ each, since PQ never contains more than c elements. Touch can be computed by a binary search of the vertical edges of D' in $O(\log d)$ time. Thus, the running time for part 2 is $O(cd \log cd)$.

Finally, part 3 takes the ordered edges of the feasible regions for quadrant pairs and finds the intersections using at most $O(1)$ operation per edge. By the observation that the feasible regions may have $O(cd)$ edges, this requires $O(cd)$ steps. □

4. The Convex/Nonconvex Case

We now consider the case of translating a convex polygon C to see if it will fit inside a simple, possibly nonconvex polygon N. We derive an algorithm to compute the boundary of $I(C,N)$. The algorithm runs in time $O((n^2+cn)\log cn)$ where n is the number of edges of N and c is the number of edges of C.

A *polygonal region* is the union of a finite number of points, (closed) line segments, and polygons. (A polygonal region is the point set of a two-dimensional simplicial complex.) The feasible region $I(C,N)$ is always a polygonal region; this follows from the material developed below. For R a polygon or polygonal region, we let $bd(R)$ denote the boundary of R. If R is a polygonal region, and $v \in R$, then v is a *vertex* if for all neighborhoods N containing v, $N \cap R$ is not N, N intersect a line, nor N intersect a half plane. Vertex $v \in R$ is *isolated* if there is a neighborhood N containing v so that $N \cap R = \{v\}$. A segment s of $bd(R)$ is *isolated* if for any interior point of s, there is a neighborhood N containing the point so that $N \cap R = N \cap l$, where l is the line extending s. See Figure 6.

The algorithms described below output the region $I(C,N)$. The representation will be a list of isolated vertices, a list of isolated line segments, and a list of polygons, where the interiors of the polygons are pairwise disjoint. We also provide for each vertex a list of incident line segments, sorted in, say, clockwise order. Note that the number of line segments of the boundary of a polygonal region is linear in the number of vertices of the region, since the graph of the boundary is planar. Hence we choose the number of vertices of a polygonal region as a measure of its size.

Suppose s is a half plane or is a segment of the boundary of a polygon, and n a unit vector pointing into the interior of the half plane or polygon. The *slope* of s is the point of tangency of s to the unit circle so that n points towards the center of the circle. Slopes p and q are *parallel* if $p = q$. If slopes p and q are not diametrically opposite, then slope r is *between p and q* if r is contained in the section of the unit circle with endpoints p and q that is smaller than 180 degrees.

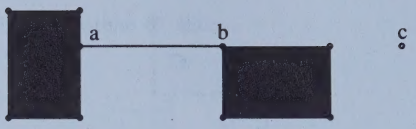


Figure 6. A polygonal region. All circled points are vertices. Vertex c is isolated, the others are not. Segment ab is also isolated.

We now begin the description of the algorithm to compute $I(C,N)$. We first consider two problems: feasibly placing C inside a convex polygon and feasibly placing C outside a set of convex polygons. To see the motivation for these problems, let H be the convex hull of N, and let T be a decomposition of the closure of $H-N$ into a set of convex polygons. We will require later that there be at most n polygons in T, each with at most n sides, and with a a total of at most $2n$ edges. Now C is feasibly placed inside N exactly if C is feasibly placed inside H and outside all the polygons in T. See Figure 7.

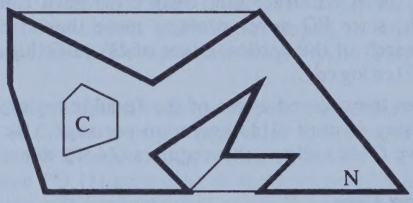


Figure 7. Light lines bound H and the polygons in T.

Proposition 4.1: $I(C,N) = I(C,H) \cap \bigcap_{D \in T} O(C,D)$.

Proposition 4.2: $I(C,N)$ is a polygonal region.

Lemma 4.3: (Chazelle[2]) Let C and D be convex polygons with c and d bounding edges, respectively. Then $I(C,D)$ is a convex region, has at most d edges, and can be found in time $O(c+d)$.

Proof: D is the intersection of a set of half planes defined by the segments of the boundary of D. C is feasibly placed inside D exactly if it is inside all of the half planes. For a particular half plane h with bounding line l, there is a point p of C that is closest to l. Note that p can always be chosen to be a vertex of C, either as the endpoint of a bounding segment of C parallel to h or as a vertex incident to two edges so that the slope of h is between the slopes of the two incident edges. Now C is in h exactly if p is in h; p is in h exactly if the reference point of C is inside the half plane parallel to h but offset by the distance between the reference point of C and p in the direction perpendicular to l. Hence C is feasibly placed inside D exactly if the reference point is within the intersection of all of the half planes constructed as described.

What is the time bound of an algorithm to compute this intersection? By a simultaneous traversal of the boundaries of C and D, the mapping from bounding half plane of D to closest vertex of C can be found in time $O(c+d)$. Computing each bounding half plane of $I(C,D)$, given a bounding half plane of D and closest vertex of C, takes constant time, for a total of $O(d)$ time for all half planes. The remaining problem is to

intersect the bounding half planes of $I(C,D)$. The usual algorithm for this problem takes time $O(d\log d)$[4]. However, the bounding half planes of $I(C,D)$ are known in slope-sorted order, since they are parallel to the edges of D, which are given in sorted order. Use of this order makes an $O(d)$ algorithm possible.

Say that a half plane is *redundant* if the intersection of two other half planes is contained in it. An equivalent condition is that a half plane is not redundant if some segment of its bounding line contributes to the boundary of $I(C,D)$. Suppose all redundant half planes have been eliminated. Then the boundary of $I(C,D)$ consists of segments of bounding lines of half planes; these segments can be determined by finding the intersection points of adjacent bounding lines in slope-sorted order. This takes time proportional to the number of half planes.

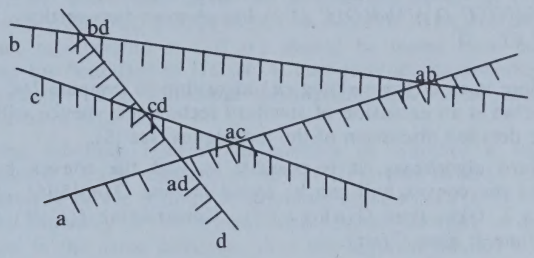


Figure 8.

How can redundant half planes be eliminated? Testing a particular triple of half planes for redundancy clearly takes constant time. We show that if there is a redundant half plane, then in fact there is a half plane contained in the intersection of its two neighbors in slope-sorted order. Using this observation it is easy to derive an $O(d)$ algorithm to eliminate redundant half planes. So suppose a, b, c, and d are half planes in slope-sorted order, and $a \cap d$ is contained in c. We show that if $b \cap d$ is not contained in c, then $a \cap c$ is contained in b. Let ad denote the intersection point of the bounding lines of a and d, similiarly for cd, bd, and ac. Clearly, $ad \in c$ iff $a \cap d \subseteq c$, and similiarly for the others. Now if $ad \in c$, $bd \notin c$, then cd lies between bd and ad along the bounding line of d. Now because the slope of c is between the slopes of b and d, ac must lie between ad and ab along the bounding line of a. Consequently, $ac \in b$, and $a \cap c \subseteq b$. See Figure 8. □

What about the feasible placements of C outside D? There are now two sets of bounding half planes. One way to understand the origin of the bounding half planes is to imagine sliding C around D, always maintaining contact. One set of half planes arises as C is slid along an edge of D; these half planes are parallel to edges of D. The second set arises as an edge of C is slid along a vertex of D; these half planes are parallel to edges of C.

Lemma 4.4: Let C and D be convex polygons with c and d bounding edges, respectively. The closure of the complement of $O(C,D)$ is a convex polygon, has at most $c+d$ edges, and can be found in time $O(c+d)$.
Proof: Similiar to the proof of the Lemma 4.3. □

Lemma 4.5: Suppose D and E are convex polygons with disjoint interiors.
(1) $bd(O(C,D)) \cap bd(O(C,E))$ is either empty, one point, two points, or a line segment.
(2) Neither $O(C,D)$ nor $O(C,E)$ is contained in the other.
Proof:

(1) A point on the boundary of $O(C,D)$ corresponds to a placement of C so that C is touching D; similiarly for $O(C,E)$. Thus a point on both boundaries correspond to a placement of C so that it touches both D and E. The set of placements so that C touches both D and E is either empty, a point, two points, or a line segment.
(2) Since D and E are disjoint, it is always possible to find a placement of D within $O(C,D)$ that overlaps E, and conversely. □

Lemma 4.6: $I(C,N)$ has at most $5(cn+n^2)$ vertices.
Proof: A vertex of $I(C,N)$ is a vertex of $I(C,H)$, a vertex of $O(C,D)$ for some $D \in T$, a vertex of $bd(I(C,H)) \cap bd(O(C,D))$ for some $D \in T$, or a vertex of $bd(O(C,D)) \cap bd(O(C,D'))$ for some $D,D' \in T$. There are at most n vertices of $I(C,H)$, and at most $n(c+n)$ vertices of $O(C,D)$ as D varies over T. By convexity, $I(C,H)$ can intersect each segment of the boundary of $O(C,D)$, for $D \in T$, at most twice, hence there are at most $2n(c+n)$ vertices of $bd(I(C,H)) \cap bd(O(C,D))$ as D varies over T. Finally, by Lemma 4.5, $bd(O(C,D)) \cap bd(O(C,D'))$ has at most two vertices, for a total of $2n^2$ for all pairs $D,D' \in T$. □

Theorem 4.7: There is an $O((cn+n^2)\log cn)$ algorithm to compute $I(C,N)$.
Proof: This algorithm is an extension of standard techniques, hence will only be sketched here. For a more detailed discussion of the techniques, see [8].

Using standard algorithms, it is possible to find the convex hull of N in time $O(n \log n)$; in fact the convex hull can be found in time $O(n)$[5][6]. Similiarly, finding the family of sets T takes time $O(n \log n)$[7]. Constructing $I(C,H)$ and $O(C,D)$, for $D \in T$ takes total time at most $O(cn)$.

It is possible to find the intersection of q line segments in time $O(q \log q + k \log q)$, where k is the total number of intersections[3]. By Lemma 4.6, there are at most $O(cn+n^2)$ intersections of the bounding segments of $I(C,H)$ and $O(C,D)$ for $D \in T$. Hence, the total time to find all intersections among the bounding segments of $I(C,H)$ and $O(C,D)$ for $D \in T$ is $O((cn+n^2)\log cn)$. Once these intersections are known, it is possible to construct an embedded planar graph G from the bounding segments of $I(C,H)$ and $O(C,D)$ for $D \in T$. G replaces each intersection point with a new vertex and breaks each intersected segment into intersection-free segments. To build this graph two data structures are needed: a segment data structure and a vertex data structure. For each segment e the segment structure stores the segments that intersect e, sorted in intersection order along e. The vertex data structure stores the segments adjacent to a vertex in clockwise order. By using a balanced tree scheme for the segment and vertex data structures, G can be constructed in time $O((cn+n^2)\log cn)$, as the number of edges is bounded by $O(cn+n^2)$.

The remaining problem is to build $I(C,N)$ from the graph G. Suppose each vertex of G has been labelled with membership in $I(C,N)$. Suppose also that each (intersection-free) segment of G has been labelled as to whether it is part of the boundary of $I(C,N)$, labelled either as an isolated line segment or as the boundary of a polygon. Then a simple search of G can be used to produce $I(C,N)$. This search must build a polygon representation from the bounding segments of the polygon, must locate isolated vertices, must combine collinear coincident segments into a common segment by deleting the common endpoint, must delete multiple parallel segments, and so forth. The time taken for the search is proportional to the number of segments and vertices, i.e. $O(cn+n^2)$.

How can the vertices and segments be labelled? A vertex is in $I(C,N)$ exactly if it is in $I(C,H)$ and in all $O(C,D)$, as D varies over T. Similiarly, a segment of G is a bounding segment of $I(C,N)$ exactly if it is in $I(C,H)$ and in all $O(C,D)$, as D varies over T. One approach would be to test each vertex and segment for membership in $I(C,H)$ and $O(C,D)$. But this would require at least cubic time, as there can be $O(n)$ sets $O(C,D)$ and $O(cn+n^2)$ segments and vertices.

A faster algorithm to infer membership in the regions $O(C,D)$ is the following. Pick

a polygon D in T. Imagine traversing the segments and vertices of G that comprise $bd(O(C,D))$. Initially membership within $O(C,D')$, for $D \neq D'$, is not known. However, as intersection points with the boundary of $O(C,D')$, for $D \neq D'$, are encountered, membership within $O(C,D')$ can be inferred. By a complete traversal of the segments comprising $O(C,D)$, complete membership information about the regions $O(C,D')$, for $D \neq D'$, can be inferred, for if there is no intersection with $bd(O(C,D'))$ for some D', then by Lemma 4.5 $bd(O(C,D)) \subseteq O(C,D')$. A second traversal of the segments and vertices comprising $bd(O(C,D))$ can be used to mark the segments that are actually in all $O(C,D')$ for $D' \in T$. It is straightforward to implement this traversal so that it runs in constant time per vertex and segment traversed. Repeating this procedure for each $D \in T$ requires a total time of $O(cn+n^2)$. A similiar traversal is needed for the segments comprising $bd(I(C,H))$.

What about membership in $I(C,H)$? The just-outlined procedure is not quite adequate, since for $D \in T$ there may never be an intersection of $bd(O(C,D))$ with the $bd(I(C,H))$, yet either $bd(O(C,D)) \subseteq I(C,H)$ or $bd(O(C,D)) \subseteq I(C,H)$ is possible. To solve this problem, membership in $I(C,H)$ should be tested once before starting the traversal. If in fact the boundary of $I(C,H)$ is encountered, membership information can be updated accordingly. The additional membership test takes time $O(n)$ for each polygon D, for a total of $O(n^2)$.

The remaining labelling information consists of whether a segment bounds a polygon, or is isolated. Each segment of $bd(I(C,H))$ and $bd(O(C,D))$ has a normal pointing into the interior of the region. A bounding segment of $I(C,N)$ is an isolated segment if it is on the boundary of two regions with normals pointing in opposite directions; if all normals point in the same direction, then the segment bounds a polygon. Clearly this part of the label can also be computed during the traversal just described. □

Clearly, it is trivial to extend Theorem 4.7 to handle the case of polygons with "holes". Each hole, decomposed into a set of convex polygons, should be added to T.

There are examples for which $I(C,D)$ has $\Omega(c,d)$ edges, where C is convex and D is nonconvex, with c and d edges, respectively. Consider Figure 3 again. Replace the steps of C by a sequence of edges, approximating a circle. Each such edge can touch each step of D on the boundary of the feasible region. Thus, the boundary has $\Omega(cd)$ edges.

Finally, we return to the inspection problem. Let n be the total number of edges in the two tolerance polygons and the polygon representing the manufactured part. Recall that our strategy was to obtain two feasible regions: one for a convex polygon placed inside a possibly nonconvex polygon, and the other for two convex polygons. From above, the former feasible region has $O(n^2)$ edges, and the latter has $O(n)$ edges. Since the latter feasible region is convex, there are at most $O(n^2)$ intersection points of the boundaries of the two regions. These can be found in $O(n^2 \log n)$ time [3]. Hence, whether the intersection of the two regions is empty can be determined in $O(n^2 \log n)$ time, solving the inspection problem.

5. Discussion

It would be interesting to find an efficient algorithm to determine whether one polygon can be translated to fit within another when both polygons are nonconvex. The example of Figure 9 shows that the boundary of the feasible region can have $\Omega(n^4)$ edges when each polygon has n edges. Thus the approach of determining the boundary of the feasible regions will be forced to use $\Omega(n^4)$ time. Can it be determined in general whether one polygon fits inside another without calculating the boundary of the feasible region? It would also be interesting to consider both rotation and translation. The work of Chazelle[2] suggest that the time bounds will probably increase substantially.

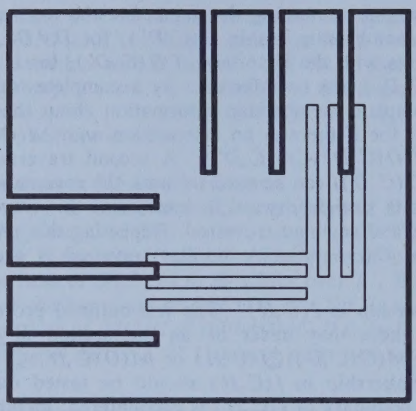


Figure 9. Feasible region with $\Omega(n^4)$ edges.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1975.

[2] B. Chazelle, "The Polygon Containment Problem," *Advances of Computing Research*, Volume 1, pp. 1-33. JAI Press, Inc.

[3] J.L. Bentley, T.A. Ottmann, "Algorithms for Reporting and Counting Geometric Intersections", *IEEE Transactions on Computers*, C-28, pp. 643-647, (1979).

[4] M.I. Shamos, "Geometric Complexity," *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*, May, 1975, pp. 224-233.

[5] R.L. Graham, F.F. Yao, "Finding the Convex Hull of a Simple Polygon," *J. Algorithms* 4, pp. 324-331 (1983).

[6] A.A. Schaffer, C.J. Van Wyk, "Convex Hulls of Piece-wise Smooth Jordan Curves," manuscript, 1984.

[7] B. Chazelle, "A Theorem on Polygon Cutting with Applications," *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, November 1982, pp. 339-349.

[8] J. Nievergelt, F.P. Preparata, "Plane-Sweep Algorithms for Intersecting Geometric Figures," *CACM* 25(10), pp. 739-747 (1982).

[9] T. Lozano-Perez, M. Wesley, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles," *CACM* 22, pp. 560-570 (1979).

DETECTING SYMMETRY IN A PLANAR FIGURE[+]

MIKHAIL ATALLAH
Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907.

Abstract

A straight line is an *axis of symmetry* of a planar figure if the figure is invariant to reflection with respect to that line. We give an $O(n\log n)$ time algorithm for enumerating all the axes of symmetry of a planar figure which is made up of (possibly intersecting) segments, circles, points, ...etc. Our algorithm is optimal, since we can establish an $\Omega(n\log n)$ time lower bound for this problem.

1. Introduction

The input to the algorithm is a description of the planar figure, and consists of a collection of linked lists, each of which contains the occurrences in the figure of one type of geometric pattern (i.e. one list contains the points, another list contains the segments, ...etc). The output produced by the algorithm is a (possibly empty) set S of straight lines each of which is an axis of symmetry of the input figure. Note that $|S| \leq n$, and that $|S| = n$ is possible. Our main result is the following:

Theorem The set S can be computed in $O(n\log n)$ time, and this is optimal.

2. Summary

In this brief summary we will give the reader the flavor of the main idea by giving an algorithm for the special case when the input figure consists only of points. The algorithm for the general case, when the figure also includes segments and other geometric patterns, is given in reference [A]. The observation that $\Omega(n\log n)$ is a lower bound can also be found in [A].

Let $p_1, \ldots, p_n$ be the input points, and let $X(p), Y(p)$ denote the coordinates of point p. Let C be the centroid of the n input points. The coordinates of point C are $X(C) = (X(p_1) + \ldots + X(p_n))/n$ and $Y(C) = (Y(p_1) + \ldots + Y(p_n))/n$, and therefore point C can be found in $O(n)$ time. Let S be the (possibly empty) set of axes of symmetry that we are trying to find. Observe that any line in S must pass through C. Note also that a reflection about any line in S must map a point whose distance from C is d into a point whose distance from C is also d. This suggests the following approach for computing S:

(i) Partition the points into equivalence classes $E_1, \ldots, E_k$ such that all the points in a class E_i are at the same distance (call it d_i) from C,

(ii) For each class E_i, find the set S_i of lines passing through C and leaving E_i invariant to reflection about them,

(iii) Compute the intersection of the S_i's (this is the desired set S).

It is clear that Steps (i) and (iii) can be done in time $O(n\log n)$. To show that Step (ii) can be done in time $O(n\log n)$, we need only show that every S_i can be computed in time $O(n_i \log n_i)$, where $n_i = |E_i|$. The algorithm for doing this follows:

Algorithm for computing S_i

[+]This work was partially supported by the Office of Naval Research under Contract N00014-84-K-0502.

Step 1: Obtain from E_i a string σ over the (infinite) alphabet consisting of the real numbers and the special symbol $\#$, as follows. Start with an empty σ and imagine an axis W revolving counterclockwise about C, starting from a position where it also passes through some point in E_i. As W revolves by 360 degrees about C, we create σ as follows. Whenever W encounters a point in E_i we do $\sigma \leftarrow \sigma\#\#$, and whenever W sweeps the angle θ separating two such encounters we do $\sigma \leftarrow \sigma\theta\theta$. Note that σ has even length. For example, the set E_i shown in Figure 1 would result in $\sigma = \#\#\theta\theta\#\#\alpha\alpha\#\#\gamma\gamma\#\#\alpha\alpha$.

Why we create σ in this way becomes clear once the following crucial observation is made: There is a one to one correspondence between the lines in S_i and the circular rotations which turn σ into a palindrome (a palindrome is a string which is equal to its reverse). For example, rotating the string σ corresponding to Figure 1 by three positions to the left turns it into the palindrome $\theta\#\#\alpha\alpha\#\#\gamma\gamma\#\#\alpha\alpha\#\#\theta$, and this corresponds to a line L in S_i (see Figure 1). This observation implies that computing S_i reduces to enumerating all the rotations which turn σ into a palindrome. This is what the next Step does.

Step 2: Compute S_i by enumerating all rotations which turn σ into a palindrome.

End of algorithm

Step 1 can clearly be implemented in time $O(n_i \log n_i)$ by sorting the points in E_i according to the order in which W encounters them when revolving about C. We now show that Step 2 can be implemented to run in $O(n_i)$ time. In what follows, $|x|$ denotes the length of a string x, and x^R denotes its reverse.

Given a string $x = a_1 \ldots a_{2m}$, rotating x by i positions to the left turns it into a palindrome if and only if $x = uu^R ww^R$ where $|u| = i$. Therefore, enumerating all the rotations which turn x into a palindrome is equivalent to enumerating all j such that each of the two strings $a_1 \ldots a_{2j}$ and $a_{2j+1} \ldots a_{2m}$ is a palindrome. This last problem is equivalent to finding all occurrences of x at even positions in $y = x^R x^R$, which can be done in time $O(|x|)$ using well-known techniques [AHU]. This shows that Step 2 can be done in time $O(n_i)$, and therefore computing S_i can be done in time $O(n_i \log n_i)$.

References

[AHU] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass.

[A] M.J. Atallah, "On Symmetry Detection," Computer Science Tech. Rept. 476 (March 1984), Purdue University, West-Lafayette, In.

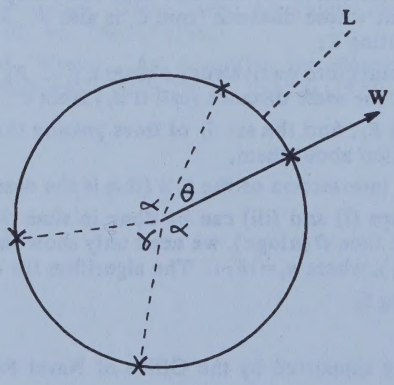


Figure 1: The points of E_i are on a circle centered at C. W rotates counterclockwise about C.

SHORTEST PATHS BETWEEN SOURCE AND GOAL POINTS LOCATED ON/AROUND A CONVEX POLYHEDRON

W. Randolph Franklin and Varol Akman
Electrical, Computer, and Systems Engineering Dept.
Rensselaer Polytechnic Institute
Troy, New York 12181

ABSTRACT

Determining the minimal free paths between two given points inside a 3-D workspace cluttered with rigid polyhedra is an outstanding open problem in Robotics. In this paper, we solve three simple instances of this problem which is commonly known as "Findpath." Our instances always treat the case of a single convex polyhedron with two points located on/around it. The solution of the third instance consists of an extension of Voronoi diagrams to the surface of a polyhedron. Thus, given several fixed points on the polyhedron, we wish to preprocess the surface so that we can find both the closest fixed point, and the minimal path, to a new point specified on the surface. This extension arises naturally from a consideration of minimal paths in 3-D where there are many polyhedral obstacles. Such a path will travel alternately through free space, and along a polyhedron's surface.

1. PROBLEM STATEMENT

A regular duty of a robot arm inside a 3-D "workspace" W (an imaginary bounding box in which several polyhedral objects reside) is to transfer, if possible, a given polyhedron from a given location to another while evading clashes with the other polyhedra (commonly termed as "obstacles"). This is known in Artificial Intelligence as the "Findpath" problem [Brooks 1983]. In this paper we solve three very specific instances of Findpath which currently stands as a very hard problem to conquer in its full generality [Schwartz and Sharir 1983; Hopcroft et al. 1984].

For the instances we treat here, we will compute the minimal (under the Euclidean metric) "free" (obstacle-avoiding) paths between two points assuming that only one point (normally chosen as the fingertip of the robot arm) is being moved in the presence of only one convex obstacle. It is noted that there exists a well-known method to shrink a moved (strictly speaking, translated) object to a point while simultaneously enlarging all the obstacles in W [Lozano-Perez and Wesley 1979; Lozano-Perez 1983]. Thus a reduction of a "Polyhedron Findpath" problem to a "Point Findpath" problem of the sort presented here is validated for a broad class of practical problems.

Recently, there has been much work on related problems in the computational geometry of motion-planning in addition to those mentioned above. The reader is referred to [Akman 1984] for a relatively rich set of references. Among these, especially [Donald 1983; O'Dunlaing and Yap 1983; Spirakis and Yap 1983; Sharir and Schorr 1984] are relevant to our work. Sharir and Schorr's work is especially interesting in that it mentions many useful results on the nature of minimal paths on a convex polyhedron. The crux of their paper is the following result

which is arrived after employing complex (and thus hard to program) data structures and algorithms:

> "Given a convex polyhedron P with n vertices and a point s on it, P can be preprocessed in $O(n^3 \log n)$ time to produce a data structure (taking $O(n^2)$ space) with the help of which one can find in O(n) time the minimal path along the surface of P from s to any other specified point g."

2. INSTANCES

Let P be a convex polyhedron and s and g ("source" and "goal") two points in the 3-D Euclidean space. Assume that $s, g \in Ext(P) \cup On(P)$ throughout this paper. (On(P), Int(P), and Ext(P) denote the surface, the interior points, and the exterior points of P, respectively.) In the sequel, we first present algorithms to compute the minimum-length s-to-g (or vice versa) paths which do not interfere with P (i.e., at most touch P but never intersect it) for the following two instances of Findpath:

> [FP1] $s, g \in On(P)$ and both points are given; and
> [FP2] s and/or $g \in Ext(P)$ and both points are given.

Finally, as a third and more interesting instance, we solve the following "Single Source-Many Goals" version of Findpath. Let s be fixed and g be varied inside W. This corresponds to a manipulator consuming a pile of objects by continuously moving them to different locations inside W. (The symmetric case of a manipulator picking up objects from several piles in order to assemble them in a fixed location is transformable to this instance by simply trading the roles of s and g.) Assume that s and the particular face F_g of P on which g is guaranteed to be located are given. (It is emphasized that the exact location of g on F_g is not known yet.) Clearly, the aim is to do some preprocessing using this limited information (i.e., P, s, and F_g) so as to compute the s-to-g minimal paths efficiently for specified "query" points which will generically be denoted by g. In our terminology:

> [FP3] $s, g \in On(P)$; s is given while g is not (but F_g, the face of P that g belongs, is known). We are allowed to preprocess to quickly answer later queries that specify many $g \in F_g$ and ask for the s-to-g minimal paths.

3. ALGORITHMS

In the following, α_0, α_1, and α_2 denote the number of vertices, edges, and faces of P, respectively. We reserve the terms "edge," "vertex," and "path" only for polygons and polyhedra; for graphs we use the less common terms "link," "node," and "sequence" in order to avoid potential confusions.

3.1. Solution of FP1

Throughout this paper, we assume that the line segment [sg] is not the shortest path since there is a very fast algorithm to detect this, that is, to intersect a line segment with a convex polyhedron P [Chazelle

1980]. This algorithm works in time $O(\log^2 \alpha_0)$.

In the following, we summarize a simple but useful fact about the shortest paths on a dihedral angle.

Lemma 1.1 (Dihedral Principle): If s and g are on different faces of a dihedral angle D, then the shortest path between them is [sx] U [xg] where x is a point on the common line cc' of the faces such that $\langle sxc = \langle c'xg$ where "$\langle$" denotes a planar angle.
Proof: Trivial. Algorithmically, x is found as follows. Rotate g about cc' until it touches to the plane of s (but specifically to the half-plane (described by cc') not including s). Thus g is transformed to g'. This is called a "development" of D into the plane. Draw [sg'] and intersect it with cc' to find x. When we fold the faces back to their original position the minimal path is placed into three dimensions on D again. □

To solve FP1 we will develop a partly geometric and partly graph-theoretic machinery. By its definition, a polyhedron is a figure in space formed by a finite number of polygons situated in a certain mutual relationship. This can be represented by a system of polygons ("evolvent" or "planar polygonal schema") in the plane [Frechet and Fan 1967].

Given a polyhedron P, we associate a polygon with each of its faces, this polygon being subject to the single condition of having the same metrical form as the associated face of P. We thus obtain a finite number of polygons in the plane each of which is separated from the others. We will couple the edges of these polygons in pairs in such a way that two "coupled" edges in the plane lead to the same edge of P. Now denote each coupled edges in the plane by the same letter. This does not yet suffice to describe the mutual relationship of the faces of P. To make it precise, each edge in the plane must be oriented by placing an "arrow" on it in an arbitrary sense, except that for two coupled edges the heads of the two arrows on them coincide when they are identified in forming the corresponding edge of P.

The evolvent Δ of P, obtained as described above, defines the intrinsic geometry of P in the sense that for a given Δ it is possible to determine the minimal paths between two points on P [Aleksandrov 1958]. In our solution of FP1, Δ is the main geometric tool. It is noted that Δ satisfies the following conditions:

i) the coupling ("gluing" in Aleksandrov's terminology) of polygons may occur either at edges or vertices;
ii) if two polygons F_i and F_j are glued at vertex v, then they are either glued together along edges containing v, or there exists a sequence of polygons, starting at F_i and ending at F_j, which are glued to one another along edges containing v; and
iii) each edge is glued to exactly one edge.

In addition to Δ, we need another simple tool. Define the "face adjacency" (or shortly, face) graph $\Omega = (V, E)$ as an undirected graph with unit links where $V = \{v \mid F_v$ is a face of P$\}$ and $E = \{(v_1, v_2) \mid F_{v1}$ and F_{v2} are adjacent along an edge of P$\}$. Now let t be a minimal

path on P from s to g. The faces that t touches while going from s to g define the "face visit sequence," $F_s, \ldots, F_g$, where each face in this sequence is adjacent to (shares an edge with) its preceding and following neighbors.

Lemma 1.2: A face visit sequence in Ω associated with a minimal path on P is "simple" (loop-free).
Proof: After recalling the fact that each face of P is also convex, one would further shorten the portion(s) of the path corresponding to the cyclic sequence(s). But then the given path is not minimal. □

Lemma 1.3: There may be O(n!) simple sequences between two specified nodes of an undirected graph with n nodes.
Proof: In the worst-case the graph is complete. (But see the argument below.) □

The number of links in Ω is much less than the number of links in a complete graph with the same number of nodes. This is due to the observation that the surface of a convex polyhedron has the structure of a planar graph; hence it can have only a linearly growing number of links in terms of its nodes. This O(n) factor of reduction in the number of links renders less freedom in moving from one node to another and helps reduce the number of simple sequences considerably.

Before we present our algorithm for FP1 we emphasize that Δ is a geometric data structure. Hence it is possible, for instance, to obtain from Δ such information as "F_1 and F_4 share edge e," or "F_6 and F_3 are disjoint." With these examples, a convenient internal representation for storing an evolvent should be self-evident (cf. [Abelson and DiSessa 1982], for example). On the other hand, Ω contains only topological information; that is, it is less informative of shape.

After above preparations we can now present our algorithm for FP1:

ALGORITHM A1.
1. Given s and g, find the faces F_s and F_g holding them. Assume that they are different; otherwise we are finished with path [sg].
2. Mark these two faces in Ω and enumerate all simple sequences between these nodes in any order.
3. For each sequence (which may be thought as a face visit sequence) compute the associated development of the relevant faces. This is accomplished using the information provided by Δ.
4. For each development compute the minimal path between the images of s and g and finally choose the smallest one(s) among them.
END A1.

In Figures 1, 2, and 3 we demonstrate this algorithm on a cube. Figure 1 depicts the evolvent and the face graph of the given cube. Figure 2 lists first few simple face visit sequences. Finally, Figure 3 shows the corresponding developments for the sequences given in Figure 2.

To the best of our knowledge, counting the number of simple sequences between two nodes of a graph is a difficult problem. This is due to the fact that given graph G = (V, E), $|V| = n$, $|E| = m$, length $l(e) \in Z^+$ for

each e ε E, specified nodes s, g ε V, and positive integers B and K, it is stated in [Garey and Johnson 1979] that the problem "Are there K or more distinct simple paths from s to g in G, each having total length B or less?" is NP-hard. Although this problem is not known to be in NP, the corresponding enumeration problem is #P-complete [Valiant 1979]. It is emphasized that #P-completeness still holds even when G is planar.

In [Yen 1971], an $O(Kn^3)$ time algorithm is given to enumerate the sequences in increasing path length. Recently, [Katoh et al. 1982] gave a new improved algorithm which works in time $O(Kc(n, m))$ if the shortest sequences from one node to all the other nodes are obtained in $c(n, m)$ time. Since $c(n, m)$ is at most $\min(O(n^2), O(m \log n))$ this algorithm is more efficient than Yen's.

3.2. Solution of FP2

When s and/or g are outside a convex polyhedron we obtain the instance FP2. In this case, it is still possible to apply the method summarized in Section 3.1 but only after a simple transformation of P to another polyhedron P' which now has s and g necessarily on its surface. The transformation step is given as follows:

ALGORITHM A2.
1. Compute the two visible outlines (silhouettes) Z_s and Z_g of P from the viewpoints s and g, respectively. These are generally nonplanar polygons which have as edges some edges of P.
2. Construct a new polyhedron P' which is the combination of three polyhedra, $P' = Q_s \cup P \cup Q_g$. Here Q_s is the pyramid having s as apex and Z_s as its base outline, and Q_g is the pyramid having g as apex and Z_g as its base outline. Note that these pyramids have generally concave bases consisting of the visible faces of P from s and g, respectively.
END A2.

The correctness of Algorithm A2 follows from the following properties:

Lemma 2.1: The respective planar projections of Z_s and Z_g from s and g are convex polygons.
Proof: Due to fact that a convex polyhedron casts a convex shadow when illuminated by a point source. □

Lemma 2.2: P' is also a convex polyhedron.
Proof: Take two points x and y inside P'. If $x \in \text{Int}(Q_s)$ and $y \in \text{Int}(P)$ then [xy] is always inside $Q_s \cup P$ for y will always be inside the infinite pyramid originating from s. A similar argument shows that the union of Q_g and P is also convex. The only unproved case is when $x \in Q_s$ and $y \in Q_g$. Let us check the dihedral angles of P' where Q_s and P, and Q_g and P are joined. Clearly, those angles are all nonreflex implying that [xy] must be totally contained inside P'. □

The following result shows that after the problem-reduction step (i.e., the computation of P' from P) the complexity of this instance is the same as with FP1.

Lemma 2.3: $\text{Size}(P') = O(\text{Size}(P))$.

Proof: The "size" of a polyhedron can be defined as the number of its vertices, edges, or faces depending on choice. In the case of a convex polyhedron they are all asymptotically equivalent since a convex polyhedron can be embedded in the plane as a graph and since α_1 is at most equal to $3\alpha_0 - 6$ in a planar graph. Hence, choose α_1 as the size measure. From Lemma 2.2, Q_s and Q_g may contribute $O(\alpha_1)$ new edges in the worst-case. □

How fast can we compute Z_s and Z_g? Below we summarize the computation of Z_s; Z_g is computed similarly. Find a point c_i inside every face of P. Compute, using Chazelle's method, the intersections of P and $[sc_i]$, $i = 1, \alpha_2$. If the intersection corresponding to a particular c_i is null then F_i is visible from s; else it is obstructed by other faces of P. In this manner all visible faces of P from s are found. This clearly takes $O(\alpha_2 \log^2 \alpha_0)$, or equivalently $O(\alpha_0 \log^2 \alpha_0)$ time. Make a list of the edges of these visible faces to compute Z_s. Any edge which appears twice in this list cannot belong to Z_s. Thus, the edges of Z_s are determined in $O(\alpha_0 \log \alpha_0)$ time via sorting this list and eliminating both elements of duplicate pairs. To get Z_s with ordered edges from this unordered set, take any point q inside any of the visible faces and sort the vertices of this set of edges anglewise about q. As soon as we find Z_s, we know all the faces of Q_s. Once we have Q_s and Q_g, P' can be easily constructed (Figure 4).

3.3. Solution of FP3

To solve FP3, we follow a Voronoi-based strategy [Franklin 1982; Franklin et al. 1983; Guibas and Stolfi 1983]. We want to partition the face F_g into subregions via a planar subdivision made of straight edges such that if a query point g is discovered to be belonging to a particular subregion of this subdivision, it must be possible to list the face visit sequence that should be taken by an s-to-g minimal path. The following algorithm accomplishes this:

ALGORITHM A3.
I. PREPROCESSING:
1. Develop onto the plane of F_g all simple visit sequences in Ω between nodes F_s (the face of P holding s) and F_g using Δ. At the end of this process, one obtains, using the Dihedral Principle, a set of points in the plane which are the "images" of s under given visit sequences.
2. Store with each image point, the visit sequence to arrive it.
3. Using standard algorithms construct the Voronoi diagram V which has these images as Voronoi centers [Guibas and Stolfi 1983].
4. "Clip" V against F_g. This simply means finding the subset V' of V included within the polygon F_g.
II. QUERYING:
1. Given a new query point g, first make sure that $g \in F_g$.
2. Using standard point location algorithms locate the subregion R including g in subdivision V' [Lee and Preparata 1977].
3. Using the stored sequence in the Voronoi center corresponding to R develop the faces specified by this sequence and find the minimal path with straightforward application of the Dihedral Principle.
END A3.

Once we preprocess the given scene for a given s and F_g, for each goal g we must determine which region of the subdivision V' it belongs to. Assume that V' has y edges. In [Lee and Preparata 1977], an algorithm is given which, after O(y log y) preprocessing time, requires $O(\log^2 y)$ search time. A faster algorithm that is based on triangulating the planar graph has the same preprocessing time but O(log y) search time [Kirkpatrick 1983].

The size y of V' depends on two things: the number of simple visit sequences between F_s and F_g in Ω, and the shape and relative location of F_g. If F_g is very big, then it will probably include most of V. If it is almost coincident with the most crowded parts of V then V' will again have many subregions. Similarly, if there are many visit sequences then the Voronoi diagram V will have many edges. In general, the complexity of constructing V on the surface of P for many specified source points will depend on the average number of regions that each face is partitioned into.

In Figure 5, we demonstrate this preprocessing method for a cube. The lists written next to each image of s in this figure specify the order of unfolding applied to arrive that point. Not all images of s are shown. Notice that if g is found to be inside the vertically hatched subregion then the minimal path is via the unfolding of simple sequence (3,2,6). On the other hand, it is via the unfolding of (3,6) when g is inside the horizontally hatched subregion.

A dynamizing technique which would allow good insertion time for an incoming point in the Voronoi diagram proves very useful in this context. Thus, we start with only a few image points (commonly corresponding to the first few short simple sequences in the face graph) and work our way to an "incomplete" Voronoi diagram easily. When we want to see the effect of another image point (not yet tried) we use a dynamic data structure to insert it. In [Gowda et al. 1983], it is proved that this is achievable in O(y) time if the Voronoi diagram has y points prior to insertion.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grants No. ECS-8021504 and ECS-8351942.

REFERENCES

ABELSON, H., AND DISESSA, A. 1982. Turtle Geometry: The Computer as a Medium for Exploring Mathematics, MIT Press, Cambridge, Mass.

AKMAN, V. 1984. Findminpath algorithms for task-level (model-based) robot programming. Manuscript, Electrical, Computer, and Systems Eng. Dep., Rensselaer Polytechnic Inst., Troy, N.Y. (Mar.)

ALEKSANDROV, A. D. 1958. Konvexe Polyeder (German, translated from Russian), Akademie-Verlag, Berlin.

BROOKS, R. A. 1983. Solving the Findpath problem by good representation of free space. IEEE Trans. on Systems, Man, and Cybernetics 13, 3 (Mar.), 190-197.

CHAZELLE, B. M. 1980. Computational geometry and convexity. Ph.D. dissertation, Computer Science Dep., Yale Univ., New Haven, Conn.

DONALD, B. R. 1983. The Mover's problem in automated structural design. In Proc. of the Harvard Computer Graphics Conf., Cambridge, Mass. (July)

FRANKLIN, W. R. 1982. Partitioning the plane to calculate minimal paths to any goal around obstructions. Tech. Rep., Image Processing Lab, Rensselaer Polytechnic Inst., Troy, N.Y. (Nov.)

FRANKLIN, W. R., AKMAN, V., AND VERRILLI, C. 1983. Voronoi diagrams with barriers and on polyhedra. Tech. Rep., Image Processing Lab, Rensselaer Polytechnic Inst., Troy, N.Y. (Dec.)

FRECHET, M., AND FAN, K. 1967. Initiation to Combinatorial Topology (translated from French), Prindle, Weber, and Schmidt, Inc., Complementary Series in Mathematics, Vol. 7, Boston, Mass.

GAREY, M. R., AND JOHNSON, D. S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, San Fransisco, Calif.

GOWDA, I. G., KIRKPATRICK, D. G., LEE, D. T., AND NAAMAD, A. 1983. Dynamic Voronoi diagrams. IEEE Trans. on Information Theory 29, 5 (Sep.), 724-731.

GUIBAS, L., AND STOLFI, J. 1983. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. In Proc. of the 15th ACM Symp. on the Theory of Computing (Apr.), 221-234.

HOPCROFT, J. E., SCHWARTZ, J. T., AND SHARIR M. 1984. On the complexity of motion planning for multiple independent objects: PSPACE hardness of the "Warehouseman's Problem." Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York (Feb.)

KATOH, N., IBARAKI, T., AND MINE, H. 1982. An efficient algorithm for k shortest simple paths. Networks 12, 411-427.

KIRKPATRICK, D. G. 1983. Optimal search in planar subdivisions. SIAM Journal on Computing 12, 1 (Feb.), 28-35.

LEE, D. T., AND PREPARATA, F. P. 1977. Location of a point in a planar subdivision and its applications. SIAM Journal on Computing 6, 3 (Sep.), 594-606.

LOZANO-PEREZ, T. 1983. Spatial planning, a configuration space approach. IEEE Trans. on Computers 32, 2 (Feb.), 108-120.

LOZANO-PEREZ, T., AND WESLEY, M. A. 1979. An algorithm for planning collision-free paths among polyhedral objects. Comm. of the ACM 22, 10 (Oct.), 560-570.

O'DUNLAING, C., AND YAP, C. K. 1983. The Voronoi diagram method of motion planning I: the case of a disc. Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York.

SCHWARTZ, J. T., AND SHARIR, M. 1983. On the Piano Mover's problem I: the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. Comm. on Pure and Applied Mathematics 36, 3 (May), 345-398.

SHARIR, M., AND SCHORR, A. 1984. On shortest paths in polyhedral spaces. In Proc. of the 16th ACM Symp. on the Theory of Computing, 144-153.

SPIRAKIS, P., AND YAP, C. K. 1983. On the combinatorial complexity of motion coordination. Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York (Apr.)

VALIANT, L. G. 1979. The complexity of enumeration and reliability problems. SIAM Journal on Computing 8, 3 (Aug.), 410-421.

YEN, J. Y. 1971. Finding the k shortest loopless paths in a network. Management Science 17, 712-716.

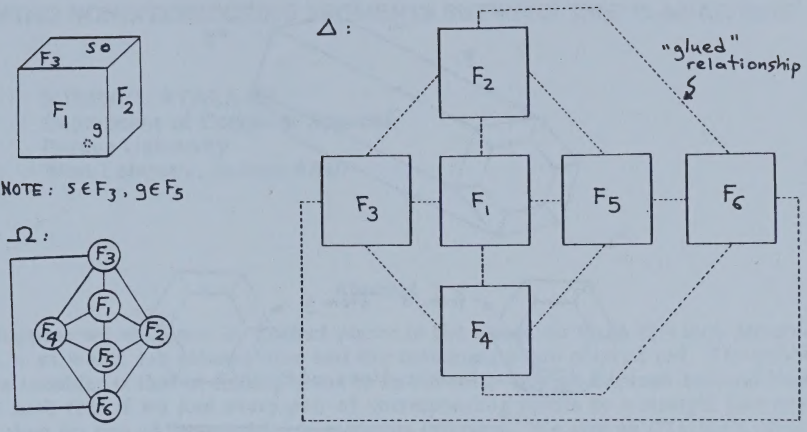


Figure 1

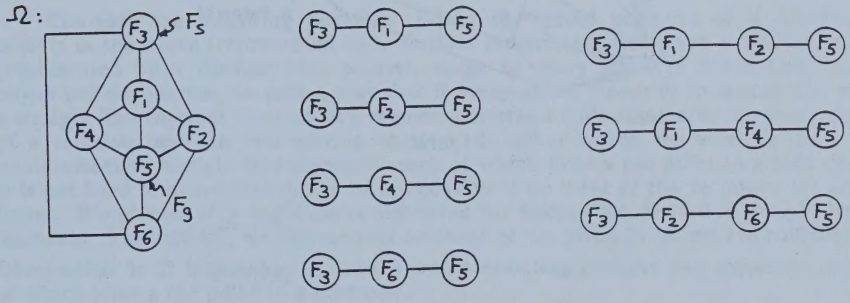


Figure 2

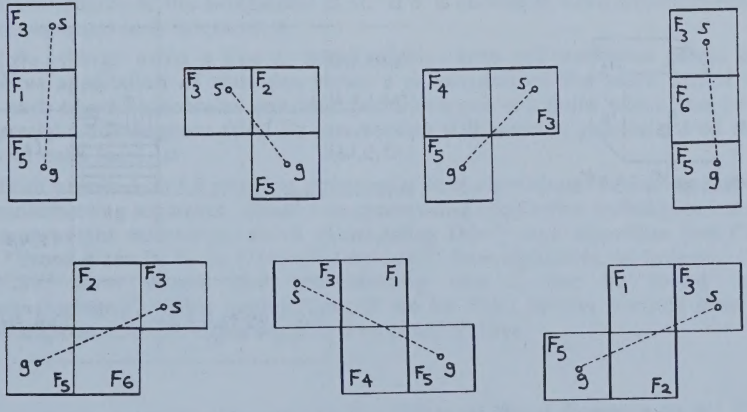


Figure 3

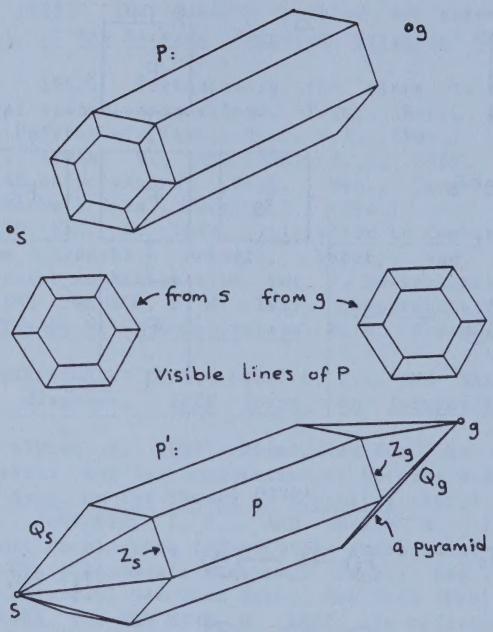


Figure 4

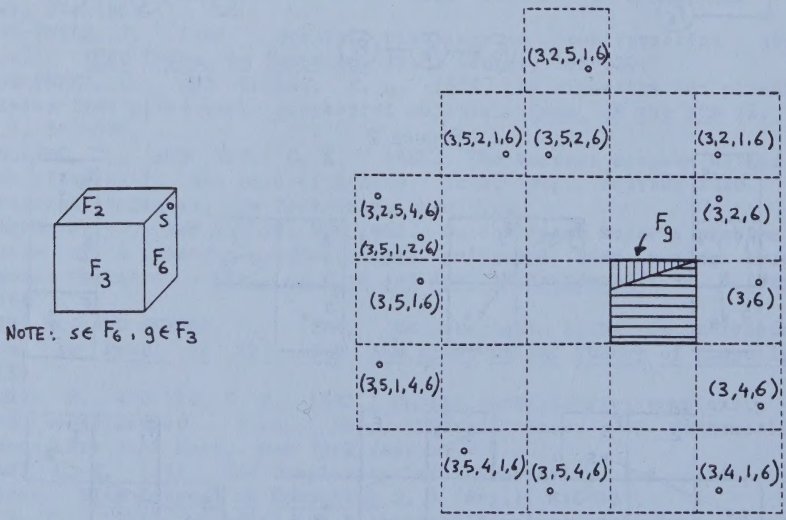


Figure 5

DRAWING NONINTERSECTING SEGMENTS BETWEEN TWO PLANAR SETS†

MIKHAIL ATALLAH
Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907.

Abstract

Suppose we are given $2n$ distinct points in the plane, no three of which are collinear, n of which are colored blue and the remaining n are colored red. The problem we consider is that of finding a one to one correspondence between red and blue points such that if we join every pair of corresponding points by a straight line segment, then no two of the resulting n segments intersect. We give an $O(n \log^2 n)$ time algorithm for computing the desired one to one correspondence.

1. Introduction and Preliminaries

Consider the following problem: Given the initial positions of n identical objects in the plane (represented by n distinct red points), and given n destinations (represented by n distinct blue points), assign to every object a destination, one object per destination, in such a way that if every object moves to its destination on a straight line segment then no two segments intersect (this eliminates the possibility of a collision between two moving objects). In other words, we want to draw n nonintersecting straight line segments, each of which joins a red point to a blue one. It is not hard to prove that this is always possible if no three of the $2n$ points are collinear. We give an $O(n \log^2 n)$ time algorithm for finding the desired n straight line segments. Throughout, we assume that no three of the given $2n$ points are collinear.

Observation 1: It is possible to draw n nonintersecting straight line segments, each of which joins a red point to a blue one.

Proof A: Let S be a set of n segments each of which joins a red point to a blue one and no two of which have a common endpoint. Let the *cost* of S be the sum of the Euclidean lengths of the n segments in it. If S is chosen to have minimum cost, then no two segments in S intersect. □

Proof B: There exists a line L which divides both red and blue points in half. Recursive application of this idea yields a subdivision of the plane into n convex faces each of which contains one blue point and one red point which can be joined by a straight line segment (that no intersection will occur is guaranteed by the convexity of each face). □

Both proofs A and B result in polynomial time algorithms for finding the desired n nonintersecting segments. Proof A suggests using the known techniques for finding minimum-weight matchings, which result in an $O(n^3)$ time algorithm (see Ch.11 in [PS]). Proof B results in an $O(n \log^2 n (\log\log n)^2)$ time algorithm, as follows. Cole et. al. [CSY] have shown that the dividing line L can be found in time $O(n \log n (\log\log n)^2)$. This implies that, if we let $T(n)$ be the running time of the divide-and-conquer algorithm suggested by proof B, then

† This work was partially supported by the Office of Naval Research under Contract N00014-84-K-0502.

$$T(n) \leq 2T(n/2) + cn\log n(\log\log n)^2$$

This implies that $T(n)=O(n\log^2 n(\log\log n)^2)$.

The next Section will present the main result of this paper: An $O(n\log^2 n)$ time algorithm. The rest of this section presents the ingredients which will be used in the algorithm.

Overmars and Van Leeuwen have designed a data structure (they call it an *augmented tree structure*) for storing n points according to (say) their x-coordinate, in such a way that the convex hull is stored at the root of the structure and is maintained as points are inserted/deleted at a cost of $O(\log^2 n)$ time per insertion/deletion (see reference [OV] for the details of how this interesting data structure functions). They also show that, if the points in two distinct augmented tree structures T_1 and T_2 are separable by a line L parallel to the y-axis, then a tangent PQ common to both the convex hull of the points in T_1 and to that of the points in T_2 (as in Fig. 1) can be found in time $O(\log n)$ (Theorem 3.2 in reference [OV]).

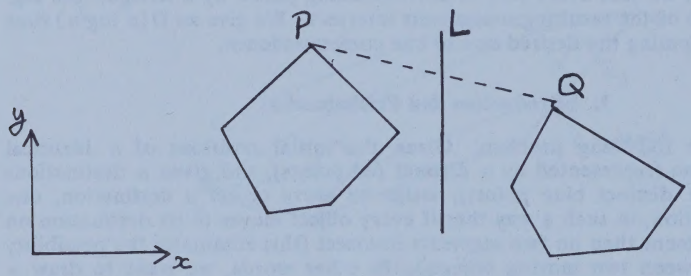


Figure 1

If G and H are convex polygons, then we say that they are *disjoint* if there is a line which separates them. We say that G and H *intersect* only if their boundaries intersect. If they are not disjoint and do not intersect then one of them contains the other. Note that we do not consider the case when G is in the interior of H as an intersection. Chazelle and Dobkin [CD] have shown that it is possible to find the intersection of a line and a convex polygon in time $O(\log n)$, and that detecting whether two convex polygons are disjoint can also be done in time $O(\log n)$.

At some point during the algorithm, we will be faced with the following problem. We will have two convex m-gons G and H that intersect and have the property that the leftmost point of G (call it A) is to the left of the lefmost point of H (call it V), while the rightmost point of H (call it W)is to the right of the rightmost point of G (call it B). See Figure 2. The following Lemma is crucial to our algorithm.

Lemma 2 A common tangent to G and H can be found in $O(\log^2 m)$ time.

Proof : We cannot immediately make use of Overmars and Van Leeuwen's already mentioned result for tangent determination, because G and H intersect. Actually, the general problem of finding a common tangent (or reporting that none exists) for two possibly intersecting convex m-gons can easily be shown to require time $\Omega(m)$ in the worst case. However, here we have the additional information about the ordering of A, B, V, W, and we now exploit this fact in order to find the desired tangent in time $O(\log^2 m)$. Let C be the point of H such that AC is a supporting line of H and H is below it (Fig. 2). Similarly, D is the point of G such that WD is a supporting

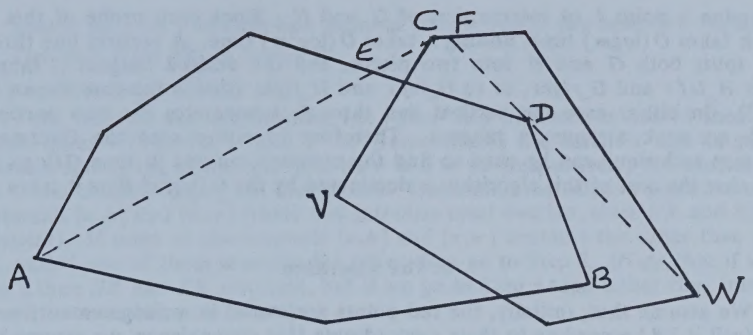


Figure 2

line of G and G is below it. Points C and D can be found in time $O(\log m)$ [S]. If line AC (WD) intersects G (H) only at A (W) then AC (WD) is the desired tangent, so assume that E (F) is another point of intersection. E and F can be found in time $O(\log m)$ [CD].

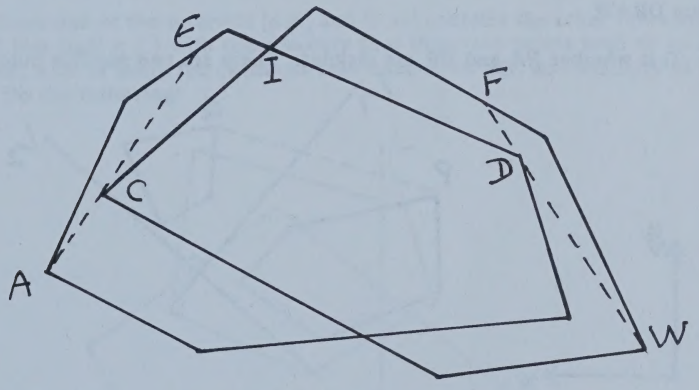


Figure 3

If E is between A and C on the line AC (as in Figure 2) then let *G_left* be the portion of G to the left of E, *H_right* be the portion of H to the right of E. The desired tangent is tangent to *G_left* and *H_right*. Since *G_left* and *H_right* are separated by the vertical line through E, Overmar's and Van Leeuwen's binary search technique can be used on *G_left* and *H_right* to find the desired common tangent in time $O(\log m)$. A symmetric argument holds when F is between W and D on the line WD. The tricky part is the case when C is between A and E *and* D is between W and F (Fig. 3). But then, in such a case, point C is in the interior of G, and since we already have W which is exterior to G, we can by a binary search

determine a point I of intersection of G and H. Since each probe of this binary search takes $O(\log m)$ time, finding I takes $O(\log^2 m)$ time. A vertical line through I then splits both G and H into two pieces, and the desired tangent is tangent to either H_left and G_right, or to G_left and H_right (this is the case shown in Figure 3). In either case the vertical line through I separates the two portions for which we seek a common tangent. Therefore in either case the Overmars-Van Leeuwen technique can be used to find the common tangent in time $O(\log m)$. It is clear that the cost of this algorithm is dominated by the $O(\log^2 m)$ time it takes to find I. □

2. The Algorithm

We assume that, initially, the red points are stored in an augmented tree structure (call it TR) according to their x-coordinate (for convenience, we assume that no two points have the same x-coordinate). As already mentioned, structure TR also contains a description of the convex hull (call it HR) of the red points, and it supports insertion and deletion operations in time $O(\log^2 n)$. We assume that a similar structure is available for the blue points (we call it TB, and we call the convex hull of the blue points HB). The algorithm we will describe assumes that TR and TB are initially available (creating them takes time $O(n \log n)$ [OV]).

We now give a recursive description of the algorithm for drawing n nonintersecting straight line segments, each of which joins a red point to a blue one.

Algorithm DRAW

Step 1: Test whether HR and HB are disjoint. There are two possible outcomes:

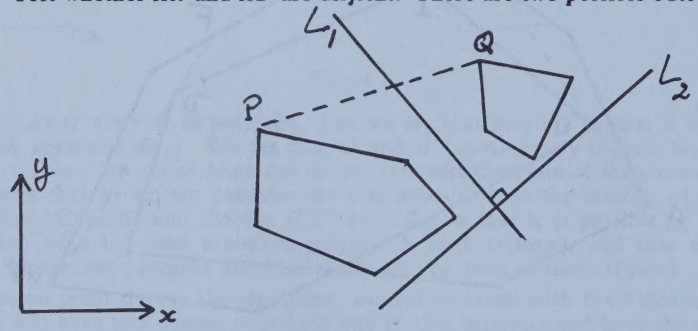


Figure 4

Case 1: HR and HB are disjoint. Find a line L_1 which separates them, then create new structures TR and TB in which the red and (respectively) blue points are stored according to their projections on an axis L_2 perpendicular to L_1 (Fig. 4), i.e. L_1 and L_2 become the new y and x axes, respectively (in the "old" TR and TB structures, the points were stored according to their projections on the old x-axis). Then repeat the following (i)-(iii) until no more points remain:

(i) Find a line tangent to both HR and HB at, say, points P and Q, respectively. It is important that both HR and HB be on the same side of the line, e.g. below it (Fig. 4).

(ii) Join P to Q by a straight-line segment (this is one of the desired n segments).

(iii) Delete P from TR, Q from TB.

When no points remain, Stop.

Case II: HR and HB are not disjoint, i.e. either they intersect or one of them contains the other. Then let a and b be (respectively) the smallest and largest x-coordinates among the red points, and let v and w be (respectively) the smallest and largest x-coordinates among the blue points. Determine the relative positions of the two intervals $[a,b]$ and $[v,w]$ (these two intervals must overlap, since HR and HB are not disjoint). If none of the intervals $[a,b]$ and $[v,w]$ contains the other then go to Step 2, and if one of them contains the other then go to Step 3. (Note that if we go to Step 2 then HR and HB intersect, but if we go to Step 3 then either they intersect or one of them contains the other.)

Step 2: Do the following:

(i) Use the algorithm described in Lemma 2 to find a common tangent to HR and HB such that HR and HB are on the same side of this tangent. Let P and Q be (respectively) the red and blue points on this tangent.

(ii) Join P and Q by a straight-line segment (this is one of the desired n segments).

(iii) Delete P from TR, Q from TB.

(iv) If TR (and hence TB) is empty then Stop. Otherwise go to Step 1.

Step 3: Since one of the intervals $[a,b]$ and $[v,w]$ contains the other there must exist a vertical line (call it L) with the property that there are points both to its left and to its right, with as many red points as blue ones to its left, and similarly to its right (Fig. 5). Do the following:

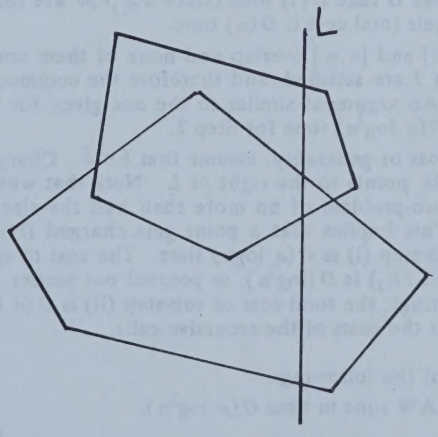


Figure 5

(i) Find the line L by performing, in parallel, two "scans" of the remaining points (both red and blue). By "in parallel" we mean that we alternate between one scan and the other, letting them progress simultaneously. One scan starts at the leftmost point and moves rightward, and the other starts at the rightmost point and moves leftward. Both scans "look" for line L until

one of them finds it.

Comment: Note that, if k red and k blue points are to the right of L, and $\hat{k}$ red and $\hat{k}$ blue points are to the left of L, then the time taken by sub-step (i) is $O(min(k,\hat{k}))$.

(ii) Find TR_1, TR_2, TB_1 and TB_2, where TR_1 and TB_1 are the augmented tree structures of the (respectively) red and blue points to the left of L, and TR_2 and TB_2 are analogously defined for points to the right of L. TR_1 and TR_2 (TB_1 and TB_2) are obtained by splitting TR (TB) about the x-component of the vertical line L.

(iii) Recursively solve each of the two subproblems consisting of the points to the left of L (using TR_1 and TB_1), and of the points to the right of L (using TR_2 and TB_2). Then Stop.

End of Algorithm Draw

Correctness of the algorithm is obvious. We now show that it runs in time $O(n \log^2 n)$. We do so by showing that the total cost of every one of Steps 1-3 is $O(n \log^2 n)$, where by "total cost" of a Step we mean its cost over all recursive calls.

Cost of Step 1: Testing whether HR and HB are disjoint and finding L_1 can be done in time $O(\log n)$ [CD], and since this is done $O(n)$ times its total cost is $O(n \log n)$ time. In Case I, since a point is involved only once in the "restructuring" of TR and TB the total cost of such restructuring is $O(n \log n)$ time. After this restructuring of TR and TB, we can use the $O(\log n)$ time algorithm of Overmars and Van Leeuwen for finding the tangent common to HR and HB, and since this is done a total of $O(n)$ times its total cost is $O(n \log n)$. On the other hand, deleting a point from TR or TB takes time $O(\log^2 n)$ and is done $O(n)$ times, and hence the total cost of such deletions is $O(n \log^2 n)$ time. The total cost of Case I is therefore $O(n \log^2 n)$ time. The tests performed in Case II take $O(1)$ time (since a, b, v, w are readily available in TR and TB), and hence their total cost is $O(n)$ time.

Cost of Step 2: Since $[a,b]$ and $[v,w]$ overlap and none of them contains the other, the conditions for Lemma 2 are satisfied, and therefore the common tangent can be found in time $O(\log^2 n)$. An argument similar to the one given for Case I of Step 1 then gives a total cost of $O(n \log^2 n)$ time for Step 2.

Cost of Step 3: Without loss of generality, assume that $k \leq \hat{k}$. Charge the $O(k)$ time it takes to find L to the $2k$ points to the right of L. Note that whenever a point is charged it ends up in a sub-problem of no more than half the size of the previous sub-problem it was in. This implies that a point gets charged $O(\log n)$ times, and hence the total cost of sub-step (i) is $O(n \log n)$ time. The cost of splitting TR (TB) into TR_1 and TR_2 (TB_1 and TB_2) is $O(\log^2 n)$, as pointed out earlier. Since there are no more than n such splittings, the total cost of sub-step (ii) is $O(n \log^2 n)$ time. We have already accounted for the costs of the recursive calls.

This completes the proof of the following:

Theorem 3 Algorithm DRAW runs in time $O(n \log^2 n)$.

3. Conclusion

Given $2n$ points, no three of which are collinear, n of which are red and the remaining n are blue, we considered the problem of finding n nonintersecting straight line segments, each of which joins a red point to a blue one. We gave an $O(n \log^2 n)$ time algorithm for this problem.

The existence of the desired nonintersecting segments can no longer be guaranteed if we drop the non-collinearity assumption, as can be seen by considering the case when all red points are on the negative part of the x-axis and all blue ones are on the positive part of the x-axis.

The following constrained versions of this problem are worth investigating:

(i) There are *obstacles* in the plane, i.e. some areas of the plane are forbidden and no segment can go through them, or

(ii) Points in a given m-subset of the red points can only be joined to points in a given m-subset of the blue ones ($m < n$).

Acknowledgement A useful conversation with Chanderjit Bajaj is gratefully acknowledged.

References

[CD] B. Chazelle and D. P. Dobkin, 'Detection Is Easier Than Computation,' *Proc. of 12th Annual ACM Symposium on Theory of Computing*, 1980, pp. 146-153.

[CSY] R. Cole, M. Sharir and C. K. Yap, "On k-Hulls and Related Problems," *Proc. 16th Annual ACM Symposium on Theory of Computing*, Washington, D.C., 1984, pp. 154-166.

[OV] M. H. Overmars and J. Van Leeuwen, 'Maintenance of Configurations in the Plane,' *Journal of Computer and System Sciences*, 1981, pp. 166-204.

[PS] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[S] M. I. Shamos, 'Computational Geometry', Dept. of Computer Sci., Yale U., New Haven, Conn., 1977.

AN ALGORITHM FOR COMPUTING ORDER k VORONOI DIAGRAMS IN THE PLANE

Binay Bhattacharya
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6

SUMMARY

The nearest neighbor problem is a very general problem that arises in several applications such as pattern recognition, density estimation, etc. This problem can be solved using nearest neighbor Voronoi diagram [1] of the designed set. The k-nearest neighbor problem is a variant of the nearest neighbor problem and can similarly be solved using order k Voronoi diagram [1]. Given a subset T of cardinality k of a set S in d-dimensional euclidean space R^d, the order k Voronoi polyhydron of T, denoted by $V_T(S)$, is defined as the locus of all points $p \in R^d$ closer to all points in T than to any other point of S not in T. The order k Voronoi diagram of S, denoted by $VD_k(S)$, is defined as the union of all such $V_{T_i}(S)$, i=1,2...,m where each T_i is a subset of S and $T_1, T_2, \ldots, T_m$ are the only k-subsets of S whose associated order k Voronoi polyhedra have non-empty interiors.

Lee [2] proposed an $O(k^2 n \log n)$ algorithm to compute the order k Voronoi diagram of a planar set of n points. His algorithm is iterative in the sense that the order (k-1) Voronoi diagram needs to be computed first before computing the order k Voronoi diagram. Eselsbrunner et al [3] presented an $O(n^{d+1})$ algorithm to compute all order k Voronoi diagram. Their algorithm is also iterative.

In this paper an algorithm to compute the order k Voronoi diagram of a planar set S of n points in $O(kn^2)$ time is presented. This algorithm is not iterative. The algorithm computes $VD_k(S)$ directly for an arbitrary value of k.

The proposed algorithm has two parts. The first part computes all the vertices of $VD_k(S)$ and establishes the adjacency relationships of the vertices. Given a Voronoi vertex of $VD_k(S)$, each of its adjacent Voronoi vertices can be computed in time proportional to the number of data points in S. Therefore the first part of the algorithm requires $O(k\ n^2)$ time in the worst case with O(kn) storage requirement since $VD_k(S)$ can have O(kn) vertices in the worst case.

Once the vertices of $VD_k(S)$ and their adjacency relationships are determined, the second part of the algorithm determines all the generating sets $T_1, T_2, \ldots, T_m$ whose associated order k Voronoi polygons

$V_{T_i}(S), i,1,2,\ldots,m$ have non-empty interiors. First the neighborhood information for each order k Voronoi polygon is determined. This is represented in the form of a graph, called neighborhood graph, denoted by NG(S), in which the vertices represent Voronoi polygons and two vertices are joined by an edge whenever the associated Voronoi polygons share an edge. An algorithm, based on plane-sweep method, to compute NG(S) in O(kn log n) time with O(kn) storage requirement has been presented. Given the NG(S) and the generating set of one order k Voronoi polygon, we can determine the generating sets of the remaining Voronoi polygons in O(kn log n) time by using the fact that for two neighboring polygons with generating sets T_1 and T_2, $(T_1-T_2)\cup(T_2-T_1)=\{p,q\}$ gives the points of S which determine the edge being shared by $V_{T_1}(S)$ and $V_{T_2}(S)$. Therefore, the second part of the algorithm requires O(kn log n) time. If we want the generating sets explicitly, $O(k^2n)$ storage spaces are required. However, using O(kn) storage space we can store NG(S) (or a spanning tree of NG(S)) and the generating set of one Voronoi polygon and obtain the generating sets whenever required in O(kn log k) time.

REFERENCES

[1] Shamos, M.I. and Hoey, D., 'Closest-point problems', Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science, pp. 151-162, 1975.

[2] Lee, D.T., 'On k-nearest neighbor problem in the plane', IEEE Trans. on Computers, Vol. C-31, No. 6, pp. 478-487, 1982.

[3] Edelsbrunner, H., O'Rourke, J. and Seidel, R., 'Constructing arrangements of lines and hyperplanes', Proceedings of 24th Symposium on Foundations of Computer Science, pp. 83-91, 1983.

AN ADAPTIVE MINIMUM-VARIANCE PREDICTOR-SMOOTHER ALGORITHM FOR IMAGE ENHANCEMENT

C. REY AND R.K. WARD, University of British Columbia, Department of Electrical Engineering, Vancouver, B.C., Canada, V6T 1W5.

ABSTRACT

A two stage adaptive filter which has good edge preservation as well as optimal smoothing characteristics is presented. The first stage predicts the location of the edges. With that knowledge the second stage adapts the appropriate piecewise linear model for the signal. It then smoothes the signal optimally to fit the model.

INTRODUCTION

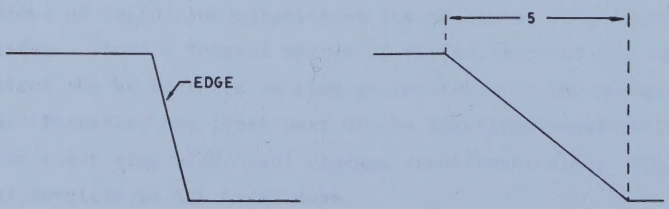
In image processing applications, the use of linear filters for smoothing has adverse effects on images. The main problem with using linear filters for smoothing is that they invariably cause smearing of image edges. Better results for edge preservation have been obtained by the use of nonlinear schemes. However, for smoothing purposes, linear filters give optimum results for "reasonably" smooth signals degraded by additive white Gaussian noise (WGN). This paper presents a filter that becomes a linear filter when the signal is "reasonably" smooth and nonlinear with good edge preservation when it is not.

The most common linear filter is the running mean filter. The output of this filter, for each pixel, is the average input intensity of a finite window of neighboring pixels.

$$\chi_o(k) = \frac{1}{w} \sum_{j=k-w/2}^{j=k+w/2} \chi_i(j)$$

$\chi_i \rightarrow$ input image
$\chi_o \rightarrow$ output image
$w \rightarrow$ window size

The effect of this filter on sharp edges is that it converts them into ramps of width w (see figures 1 and 7).



(a) Input Image (b) Output Image

Fig. 1. Degradation of a sharp edge by a running mean filter of window size 5.

If the input signal is "reasonably" smooth it can be approximated by a ramp within a small window. In this case, the running-mean filter is an optimum filter. To show this consider the model for a ramp:

$$\begin{cases} x_{k+1} = x_k + \Delta_k \\ \\ \Delta_{k+1} = \Delta_k \end{cases} \qquad (1)$$

and the measurement equation:

$$Z_k = x_k + V_k \qquad (2)$$

where V_k is a WGN sequence.

Consider the linear filter with coefficients: $a_{-w/2}, \ldots, a_0, \ldots, a_{w/2}$

Define $\underline{a} = \begin{bmatrix} a_{-w/2} \\ \vdots \\ a_0 \\ \vdots \\ a_{w/2} \end{bmatrix}$; $\underline{Z}_k = \begin{bmatrix} Z_{k-w/2} \\ \vdots \\ Z_k \\ \vdots \\ Z_{k+w/2} \end{bmatrix}$

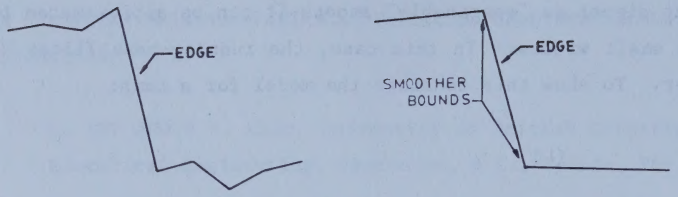
The output is given by $x_k = \underline{a}^T \underline{Z}_k$.

It can be shown that $\underline{a} = \begin{bmatrix} 1/w \\ \vdots \\ 1/w \\ \vdots \\ 1/w \end{bmatrix}$ minimizes the variance of $e_k = x_k - \hat{x}_k$

That is $\hat{x}_k(k) = 1/w \sum_{j=k-w/2}^{j=k+w/2} Z_j$ is an optimum running mean filter.

A nonlinear filter that has been used with some success for smoothing is the median filter ([1], [4]). The median filter has excellent edge preservation and also performs very well in eliminating sparse impulse noise (see fig. 8). Several schemes have combined the excellent properties of the median with a linear smoother ([3], [6]). The most successful schemes have proven to be adaptive schemes.

The algorithm presented here also uses an adaptive scheme. However, it differs from all others in that it uses an adaptive piecewise linear model. For a small window the model is composed of connected ramps that may vary in length and number. For a smooth signal, the model (inside a small window) will be a ramp; for an edge, the model (inside a small window near the edge) will be a set of three ramps. See fig. 2.



Input Signal Model of Input Signal

Fig. 2. The modelling of an edge by three ramps.

The proposed filter is composed of two stages:

(1) Prediction Stage - This stage predicts the location of the boundary points of "reasonably smooth" intervals. These locations (fig. 2) are called "smoother bounds". See fig. 6 for an example of predicted smoother bounds.

(2) Smoothing Stage - This stage uses the smoother bounds obtained above to draw a signal model inside a small sliding window (called a smoother window). The model is obtained by fitting a ramp between each pair of consecutive smoother bounds inside the smoothing window. Thus the overall model (inside the smoothing window) is a set of connected ramps. The output of each pixel is taken to be the fitted value of the window's center pixel.

THE PREDICTION STAGE

The prediction stage finds those pixels (called smoother bounds) which bound regions where the signal is "reasonably" smooth. This is done by imposing a ramp model inside a sliding prediction window. If the extrapolated intensity for the pixel following the window differs from the measured intensity by more than a certain threshold, the last pixel in the prediction window is said to be a smoother bound. The size of the prediction window is not fixed; it depends on the location of the last detected smoother bound. Following the detection of a smoother bound the prediction window size is reduced to two pixels (the smoother bound and the following pixel). Each next prediction window is obtained by adding one more pixel to the previous window. The maximum length of the prediction window, however, is fixed so as to avoid gross mismodelling errors.

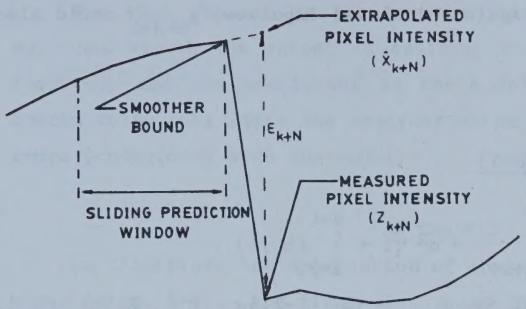


Fig. 3. The detection of a smoother bound.

The predictor is based on the model (1) and the measurement equation (2) with V_k = WGN sequence with standard deviation σ. Consider a window of size N, the predicted value of χ_{k+N} is

$$\chi_{k+N} = \underline{a}^T \underline{Z}_k$$

where $\underline{a} = [a_0, a_1, \ldots a_{N-1}]^T$

and $\underline{Z}_k = [Z_k, Z_{k+1}, \ldots, Z_{k+N-1}]^T$

The optimal $\underline{a}$ so as to minimize $E = [(\chi_{k+N} - \chi_{k+N})^2]$ is given by the formula (proven in Appendix I):

$$a_i = 2\,\frac{3i - N + 1}{N(N-1)}\,;\quad i = 0,\ldots,N-1$$

The prediction error is $e_{k+N} = (\chi_{k+N} - \chi_{k+N})$; its variance is

$$\sigma^2_{e_{k+N}} = \sigma^2 \sum_{i=0}^{N-1} a_i^2.$$

Let $\varepsilon_{k+N} = Z_{k+N} - \chi_{k+N}$ then $\varepsilon_{k+N} = V_{k+N} + e_{k+N}$. Because e_{k+N} and V_{k+N} are independent, the variance of ε_{k+N} is:

$$\sigma^2_{\varepsilon_{k+N}} = \sigma^2_{e_{k+N}} + \sigma^2_{V_{k+N}} = \sigma^2\Big(1 + \sum_{i=0}^{N-1} a^2{}_i\Big)$$

The predictor's basic function is to detect where the ramp model fails. Model failures are detected by examining ε_{k+N} --the difference between the predicted χ_{k+N} and the measured Z_{k+N}. If $|\varepsilon_{k+N}|$ exceeds a certain threshold the model is said to have failed. Currently this threshold is set to 2.5 $\sigma_{\varepsilon_{k+N}}$ which gives a 98.8% confidence interval. If $|\varepsilon_{k+N}| > 2.5\ \sigma_{\varepsilon_{k+N}}$ the pixel k+N-1 is said to be a smoother bound.

Note that there are two bounds corresponding to every edge. See fig. 2.

To provide protection against isolated impulses, χ_{k+N+1} could also be predicted linearly:

$$\chi_{k+N+1} = \underline{b}^T \underline{Z}_k$$

Now, $b_i = 2\,\dfrac{3i\,(N+3) - (N+1)(N+5)}{(N-1)\,N\,(N+1)}$

The variance of ε_{k+N+1} is $\sigma^2_{\varepsilon_{k+N+1}} = \sigma^2 \left(1 + \sum_{i=0}^{N-1} (b_i)^2\right)$.

The 98.8% confidence interval for ε_{k+N+1} is: $\left(-2.5\sigma_{\varepsilon_{k+N+1}}, 2.5\sigma_{\varepsilon_{k+N+1}}\right)$. For a bound to be detected both ε_{k+N} and ε_{k+N+1} would have to lie outside of their 98.8% confidence intervals.

Note that the prediction stage creates a list of smoother bounds (their locations only). The smoother will assume that the signal is reasonably smooth between consecutive smoother bounds.

THE SMOOTHER STAGE

The smoother uses a different sliding window (called a smoother window). Inside this window the signal is modelled by a series of connected ramps (piecewise linear model). To build the model the smoother checks for smoother bounds falling within the smoother window. Then the smoother fits (in a least mean square (LMS) sense) ramps between each pair of consecutive smoother bounds.

The output of the smoother is the fitted intensity value of the window's center pixel. This value could be shown to be (see Appendix II):

$$\chi_k = \underline{b}^T \left(\underline{B}\,\underline{B}^T\right)^{-1} \underline{B}\,\underline{Z}_k$$

where χ_k = estimate of χ_k

$$\underline{Z}_k = \begin{bmatrix} Z_{k-w/2} \\ \vdots \\ Z_k \\ \vdots \\ Z_{k+w_2} \end{bmatrix} \quad \text{and} \quad \underline{b} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix};$$

{w is the smoother window size}. The number of elements in b = number of fitted ramps + 1.

The matrix $\underline{B}$ depends only on the smoother window size and the location of the smoother bounds. Thus, $\underline{B}$ varies from one window to the next.

The size chosen for the smoother window is a parameter depending on: how smooth the output is desired; the amount of computational time required; and the confidence in the model. If the smoother window is chosen to be very large the smoother output becomes a series of connected ramps (coinciding with the model).

AN EXAMPLE

To illustrate the application of the filter presented five plots are shown (figs. 4-8). The first plot shows the original signal. The second plot shows the image degraded by additive WGN and two impulses. The third plot shows the result of applying the present algorithm to the degraded image. The algorithm used a prediction window of size 5, a smoothing window size of 7 and a confidence interval for the predictor of 98.8%. Note the excellent edge preservation and the smoothness achieved by the algorithm. The fourth plot shows the application of a running mean filter of window size 5. Note the degradation of the two image edges. The last plot shows the application of a median filter of window size 5. Note the excellent edge preservation but suboptimal smoothness.

CONCLUSION

A two-stage filter for signal smoothing has been presented. The main feature of this filter it that is uses an adaptive model for the signal. The adaptability of the model gives exceptional edge preservation while keeping the optimal smoothing properties of linear filters.

REFERENCES

[1] J.W. Tukey, "Nonlinear (non-superposable) methods for smoothing data," 1974 EASCON Conf. Rec., p. 673.

[2] A. Rosenfeld, A.C. Kak, Digital Picture Processing, Vol. 1, 2nd Ed., Academic Press, 1982.

[3] A.C. Bovic, T.S. Huang, D.C. Munson, Jr., "A generalization of median filtering using linear combinations of order statistics," IEEE Trans. Acoust. Speech, and Signal Proc., Vol. ASSP-31 pp. 1342-1350, Dec. 1983.

[4] C.L. Mallows, "Some theory of nonlinear smoothers," Annals of Statistics, Vol. 8, pp. 695-715, 1980.

[5] A. Gelb et al, Applied Optimal Estimation, M.I.T. Press, 1974.

[6] Y.H. Lee, S.A. Kassam, "Some Generalizations of Median Filters", Proc. IEEE Conf. on Acoustics, Speech and Signal Proc., pp. 411-414, 1983.

APPENDIX I

The equation of the linear extrapolator is:

$\chi_{k+N} = \underline{a}^T \underline{z}_k$.

Assuming that the model (1) is valid within the prediction window:

$z_{k+i} = \chi_k + i\Delta_k + v_{k+i}$, $i = 0,\ldots,N-1$.

Thus:

$$\underline{z}_k = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & N-1 \end{bmatrix} \begin{bmatrix} \chi_k \\ \Delta_k \end{bmatrix} + \underline{V}_k$$

In matrix form: $\underline{z}_k = \underline{B}^T\underline{\beta} + \underline{v}_k$; where $\underline{\beta} = \begin{bmatrix} \chi_k \\ \Delta_k \end{bmatrix}$ and $\underline{v}_k = [v_k,\ldots,v_{k+N-1}]^T$ (v_k is a WGN sequence with variance of σ^2).

The minimum variance unbiased estimate for β is known to be:

$\underline{\beta} = [\underline{B}\underline{B}^T]^{-1}\underline{B}\underline{Z}_k$

and $\chi_k = [1,0]\underline{\beta}$ and $\Delta_k = [0,1]\underline{\beta}$, thus $\chi_{k+N} = [1,N]\underline{\beta}$ and this gives $\underline{a} = \underline{B}^T(\underline{B}\underline{B}^T)^{-1}[1,N]$. Also, $\chi_{k+N-1} = [1,N+1]\underline{\beta}$ and $\underline{b} = \underline{B}^T(\underline{B}\underline{B}^T)^{-1}[1,N+1]$. The elements a_i of $\underline{a}$ and b_i of $\underline{b}$ could be found to be

$$a_i = 2\,\frac{3i-N+1}{N(N-1)} \qquad i = 0,\ldots,N-1$$

$$b_i = 2\,\frac{3i(N+3)-(N+1)(N+5)}{(N+1)N(N-1)} \qquad i = 0,\ldots,N-1$$

APPENDIX II

This appendix illustrates the use of the smoother bounds to obtain an adaptive smoothing of the signal. Given a pixel, the smoother considers a smoothing window centered on the pixel. A smoothed adaptive estimate of the pixel's intensity is obtained through fitting a ramp between each pair of consecutive smoother bounds (inside the smoother window). The center pixel's intensity estimate is obtained from the fitted model. This will be demonstrated via an example. From the example the general procedure may be easily deduced.

Assume the smoother has window size = 11 as shown in fig. 9. Assume that the smoother bounds have been identified, say at k-3, k+1 and k+3. Four connected ramps will be fitted and from them the smoothed value χ_k will be found. The slopes of the different ramps will be referred to as Δ_1, Δ_2, Δ_3 and Δ_4 (e.g. $\Delta_3 = \chi_{k+3} - \chi_{k+2} = \chi_{k+2} - \chi_{k+1}$). The measurement equation is:

$z_i = \chi_i + v_i$

where v_i is a WGN sequence.

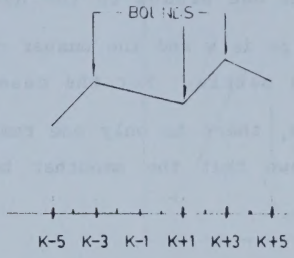


Fig. 9. Example of a four ramp model.

Let $\underline{z}_k$ be the vector of known measurements: $(z_{k-5},\ldots,z_k,\ldots z_{k+5})^T$ and let $\underline{v}_k$ be the corresponding vector of noise. The series of ramps will be modelled as:

$$\underline{z}_k = \begin{bmatrix} 1 & -2 & -3 & 0 & 0 \\ 1 & -1 & -3 & 0 & 0 \\ 1 & 0 & -3 & 0 & 0 \\ 1 & 0 & -2 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 2 & 0 \\ 1 & 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} \chi_k \\ \Delta_1 \\ \Delta_2 \\ \Delta_3 \\ \Delta_4 \end{bmatrix} + \underline{v}_k$$

which is of the form:

$$\underline{z}_k = \underline{B}^T\underline{\beta} + \underline{v}_k \text{ where } \underline{\beta} = \begin{bmatrix} \chi_k \\ \Delta_1 \\ \vdots \\ \Delta_4 \end{bmatrix} \text{ is the unknown.}$$

The minimum variance unbiased estimate of $\underline{\beta}$ is known to be:

$$\underline{\beta} = (\underline{B}\underline{B}^T)^{-1}\underline{B}\,\underline{z}_k \ .$$

The unknown quantity of interest χ_k is then:

$$\chi_k = [1,0,0,0,0,]\ \underline{\beta} \ .$$

Let $\underline{b}^T = [1,0,0,0,0]$.

Thus $\chi_k = \underline{b}^T(\underline{B}\underline{B}^T)^{-1}\underline{B}\,\underline{z}_k$.

The matrix $\underline{B}^T$ usually varies from one window to the next. Its dimensions also vary. If the window size is w and the number of bounds within it is s then $\underline{B}^T$ is a w x (s+2) matrix. For the case when no bounds exist within the smoother window, there is only one ramp in the model. In this case it could be shown that the smoother becomes a running mean filter.

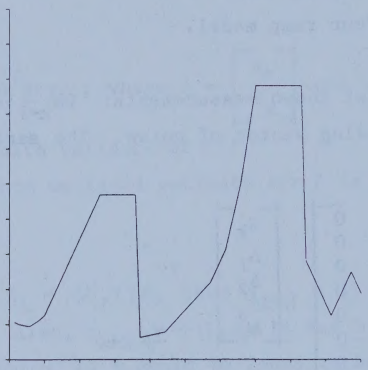


Fig 4

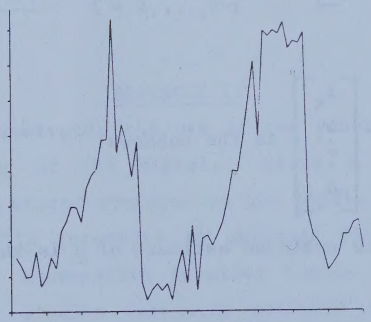


Fig 5

Figure 4. Original signal.

Figure 5. Signal degraded by additive WGN with σ = 1.2 and two impulses.

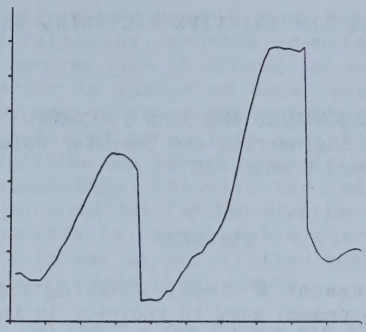


Figure 6. Result of filtering - forward predictor window size is 5, smoothing window is 7, 98.8% confidence interval used for smoother bound prediction.

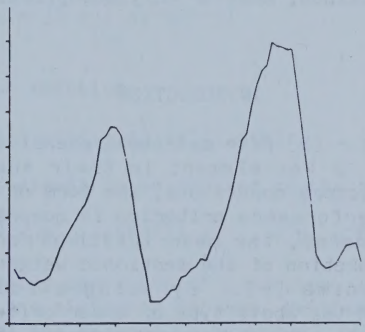


Figure 7. Running mean filter with window size 5. Note the degradation of the signal edges.

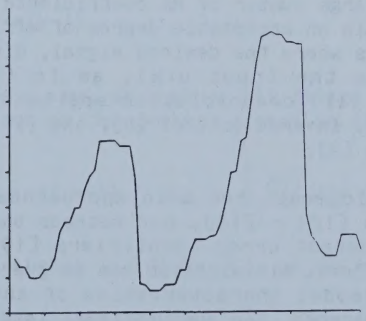


Figure 8. Median filter with window size 5.

A SUBOPTIMAL SOLUTION TO IIR ADAPTIVE FILTERING USING AN EXTENDED LMS ALGORITHM

EMMANUEL FERNANDEZ-GAUCHERAND AND JOHN Y. CHEUNG
School of Electrical Engineering and Computer Science,
University of Oklahoma, Norman, OK

ABSTRACT

In this paper, we present a simple updating algorithm for recursive adaptive filters. The approach used is indirect in that the error between the desired signal and the output of the adaptive recursive filter (the output error) is related to the error between the desired signal and the output of a two input-one output finite impulse response filter. The minimization of the former error is sought by minimizing the latter using a more general form of the least mean squared algorithm. Convergence analysis of the obtained algorithm shows that global minimization of the output error can be attained, however only suboptimal minimization will be possible in general.

INTRODUCTION

Widrow et. al. [1] - [5] have made comprehensive contributions in FIR adaptive filtering. A key element in their analyses is the linear combiner. Under stationary conditions, the form of the widely used mean-squared error (MSE) performance criterion is quadratic in the combiner's weight vector. In general, the mean-(2K)th error, K being a natural number, is a convex function of the mentioned weight vector and therefore cannot have local minima [5]. By using a local estimate for the expectation operator in the above type of error criterion, Widrow and Hoff [1] formulated their well known least mean-squared (LMS) algorithm to seek global minimization iteratively.

The use of an IIR structure could possibly reduce complexity and improve performance when compared to all-zero adaptive filters, in those applications where a large number of MA coefficients are required for the latter in order to obtain an acceptable degree of efficiency. Such is the case in those situations where the desired signal, d(k), is best described by an ARMA process on the input u(k), as in (i) a general system identification mode, (ii) deconvolution applications, e.g., channel equalization [6], [7] and inverse control [8], and (iii) in certain type of model following problems [9].

For algorithm development, two main approaches have been employed, namely descent methods [10] - [18], and methods based on hyperstability theory as applied to output error identifiers [19] - [25]. With the descent methods, only local minimization can be guaranteed [14], [16] due to the generally multimodal characteristics of the MSE surfaces being descended [17]. Furthermore, the computational load is heavily increased by the need of on-line stability monitoring and projection techniques for the auxiliary ARMA process approximating the gradients needed for the descent, as commented by Johnson [26]. On the other hand, the hyperstable family of adaptive recursive filters emerged as a result of the use of system identification formulations, for which a vast literature rich in theoretical results is available, laying down a pattern which has become the current trend [26]. However, a system identification mode of operation

must be regarded as only a less general, more structured problem than that of general adaptive filtering, and the translation of results from the former to the latter carries with it several yet unanswered questions [26]. Particularly, in order to guarantee convergence for the hyperstable filters, a priori knowledge of the characteristic polynomial of the process generating d(k) is required to satisfy a strictly positive real (SPR) condition. This condition has been a severely limiting factor to the use of this family of algorithms. Notwithstanding these advances for the hyperstable family, an algorithm for IIR adaptive filtering which exhibits low computational complexity, predictable characteristics, and simple stability conditions is yet an unfulfilled need. The algorithm that we present next tries to meet these requirements.

A TWO INPUT - ONE OUTPUT FIR SCHEME

For clarity sake, let us focus initially on a deterministic system identification scheme, where the desired signal, d(k), is assumed to be reasonably modeled by the general ARMA process given below

$$d(k) = \sum_{j=0}^{M_d} b_j\, u(k-j) + \sum_{i=1}^{N_d} a_i\, d(k-i) \qquad (1)$$

or using delay operator notation

$$d(k) = B^{0}_{Md}(q^{-1})\, u(k) + A^{1}_{Nd}(q^{-1})\, d(k) \qquad (2)$$

where u(k) is an input signal to be applied to the identifier too, and

$$B^{n}_{m}(q^{-1}) = b_n\, q^{-n} + b_{n+1}\, q^{-(n+1)} + \ldots + b_m\, q^{-m}, \qquad n \leq m \qquad (3)$$

where q^{-p} denotes a delay operation of p samples, i.e., $q^{-p}\, u(k) = u(k-p)$. Other delay operators are similarly defined. Neither the coefficients, b_j's and a_i's, nor M_d and N_d are assumed to be known.

Let $y_u(k)$ denote the response of a FIR filter of order M to the input u(k), that is

$$y_u(k) = \sum_{j=0}^{M} \hat{b}_j\, u(k-j) \qquad (4)$$

Now, let $y_d(k)$ be the output of another FIR filter of order (N-1) having a one step delayed version of the desired signal as its input, as shown below

$$y_d(k) = \sum_{p=0}^{N-1} \hat{a}_{p+1}\, d((k-1) - p) \quad = \sum_{i=1}^{N} \hat{a}_i\, d(k-i) \qquad (5)$$

Augmenting the output signal in (4) by (5), we define

$$y_a(k) = y_u(k) + y_d(k)$$

$$= \sum_{j=0}^{M} \hat{b}_j\, u(k-j) + \sum_{i=1}^{N} \hat{a}_i\, d(k-i) \qquad (6)$$

or in operator form

$$y_a(k) = \hat{B}^0_M(q^{-1})\, u(k) + \hat{A}_n(q^{-1})\, d(k) \qquad (7)$$

The error of $y_a(k)$ with respect to d(k) is given by

$$\begin{aligned} e(k) &= d(k) - y_a(k) \\ &= d(k) - \hat{B}^0_M(q^{-1})\, u(k) - \hat{A}^1_N(q^{-1})\, d(k) \\ &= [1 - \hat{A}^1_N(q^{-1})]\, d(k) - \hat{B}^0_M(q^{-1})\, u(k) \end{aligned} \qquad (8)$$

Note that (8) is linear in both $\hat{A}^1_N(q^{-1})$ and $\hat{B}^0_M(q^{-1})$, thus (7) can be viewed as a two input, u(k) and d(k-1), one output, $y_a(k)$, FIR filter. Minimizing (8), say in a MSE sense, and given that (7) has enough complexity to span all the modes of (1), that is if $M \geq M_d$ and $N \geq N_d$, then (1) can be approximated by (7). Moreover, the coefficients of (7) will constitute estimates of those of (1), where the latter processes is viewed as having $(M - M_d)$ and $(N - N_d)$ higher order MA and AR coefficients, respectively, equal to zero.

An important fact to be realized is that (8) will be linear in the coefficients of (7) regardless of the nature of d(k), and regardless whether $M \geq M_d$ and $N \geq N_d$. Hence the form of (1) is not a necessary assumption for the process d(k).

Another way to realize the FIR characteristic of (7) is to compute $y_a(k)$ by means of a linear combiner. Defining the vector of parameters as

$$\begin{aligned} \hat{\underline{\theta}} &= [\, \hat{\underline{B}} \mid \hat{\underline{A}} \,]^T \\ &= [\hat{b}_0\ \hat{b}_1\ \ldots\ \hat{b}_M \mid \hat{a}_1\ \hat{a}_2\ \ldots\ \hat{a}_N]^T \end{aligned} \qquad (9)$$

and the information vector as

$$\begin{aligned} \underline{X}(k) &= [\, \underline{U}(k) \mid \underline{D}(k-1)]^T \\ &= [u(k)\ u(k-1)\ \ldots\ u(k-M) \mid d(k-1)\ d(k-2)\ \ldots\ d(k-N)]^T \end{aligned} \qquad (10)$$

then (7) can be rewritten as

$$y_a(k) = \hat{\underline{\theta}}^T \underline{X}(k) = \underline{X}^T(k)\, \hat{\underline{\theta}} \qquad (11)$$

The structure of (11) above can be viewed as an extrapolated version of the linear combiner used by Widrow et. al., where two tap delay lines are used, instead of one, having as inputs d(k) and u(k), to form the two blocks constituting (10).

Parallel to the single tap-delay line case, the optimal vector of coefficients, denoted $\hat{\underline{\theta}}^*$, for any values of M and N and for general d(k), is given by

$$\hat{\underline{\theta}}^* = \{E[\underline{X}(k)\, \underline{X}^T(k)]\}^{-1}\, \{E[d(k)\, \underline{X}(k)]\} \qquad (12)$$

Equation (7) is sometimes called the equation error in a purely system identification setting [26] where (1) holds, which is actually an abuse since equation error formulations cover a broader scope of identification problems besides this one [27]. The applicability of (7) to more general minimization problems, as commented above, is a commonly overlooked fact. In the following discussion, the IIR adaptive filtering problem is related

to the two input - one output FIR filter scheme of (7), and an extension to the LMS algorithm is presented which is capable of seeking for the optimal set of parameters of (12) iteratively.

A CONCEPT OF ERROR BY KALMAN

Let f(k) be an arbitrary ARMA process as given below

$$f(k) = \sum_{j=0}^{M} \gamma_j \, u(k-j) + \sum_{i=1}^{N} \delta_i \, f(k-i) \tag{13}$$

or in operator notation

$$f(k) = T_M^0 (q^{-1}) \, u(k) + \Delta_N^1 (q^{-1}) \, f(k) \tag{14}$$

The time domain transfer function for the process of (14) is given by

$$\frac{f(k)}{u(k)} = \frac{T_M^0 (q^{-1})}{1 - \Delta_M^1 (q^{-1})} \tag{15}$$

Now, given a process d(·) that is arbitrarily correlated with u(·), the output error is defined as follows

$$\begin{aligned} e_1(k) &= d(k) - f(k) = d(k) - \frac{T_M^0 (q^{-1})}{1 - \Delta^1{}_N (q^{-1})} \, u(k) \\ &= \frac{[1 - \Delta^1{}_N(q^{-1})]d(k) - T_M^0(q^{-1})}{1 - \Delta_N^1 (q^{-1})} \, u(k) \end{aligned} \tag{16}$$

Note that (15) and (16) share the same characteristic polynomial. Hence, minimization of $e_1(k)$ results in a nonlinear problem in the δ_i's coefficients. However, we can see that a related error, $e_2(k)$, defined as the numerator of (16)

$$e_2(k) = [1 - \Delta_N^1(q^{-1})]d(k) - T_M^0(q^{-1}) \, u(k) \tag{17}$$

is linear in both the γ_j's and δ_i's coefficients.

The minimization of (17) is a much more amenable task than that of (16), and we may use the former as an indirect approach to the latter. Clearly, if $e_2(k)$ can be made equal to zero, $e_1(k)$ will go to zero as well, since (16) results for passing $e_2(k)$ through an AR filter with denominator $1-\Delta_N^1(q^{-1})$. Similar filtering of a linear-in-the-coefficients error is found in current IIR adaptive filtering techniques [26]. This relation between (16) and (17) has been known for some years to the system identification community, specially as related to batch processing techniques [28], [29] and was first suggested implicitly by Kalman in 1958 [30]. Sometimes, (17) is called the equation error. For the sake of avoiding confusion, we will refer to it hereinafter as the Kalman error. Note again that no particular a priori assumptions about the nature of the correlation between d(·) and u(·) is needed.

THE EXTENDED LMS (ELMS) ALGORITHM

If batch processing techniques are used to minimize (17), say in a least squares sense, all that is needed is to collect sufficient measurements of $d(\cdot)$ and $u(\cdot)$ such that a nonsingular set of normal equations can be stated [29]. However, if the MSE iterative minimization of (17) is desired, for the adaptive version of (13), the problem that (17) is not directly measurable from access to $d(k)$ and $f(k)$ is encountered, but (17) is needed to estimate the performance of the minimization. Comparing (17) and (8) we realize that these two equations have identical form, hence (17) can be generated by use of an auxiliary adaptive filter of similar structure as that of (7). With this in mind, the method of steepest descent can be used, such that (12) can be sought iteratively.

Parallel to [1], the following approximation is done

$$E[(e^2(k))^2] \triangleq (e_2(k))^2 \tag{18}$$

which is used in the computation of the needed gradients

$$\nabla_{\gamma_j} \{E[(e_2(k))^2]\} \triangleq \nabla_{\gamma_j} \{(e_2(k))^2\}$$

$$= -2\, e_2(k)\, u(k)\, q^{-j}$$

$$= -2\, e_2(k)\, u(k-j) \; ; \quad j = 0, 1, \ldots, M \tag{19a}$$

$$\nabla_{\delta_i} \{E[(e_2(k))^2]\} \triangleq \nabla_{\delta_i} \{(e_2(k))^2\}$$

$$= -2\, e_2(k)\, d(k) q^{-i}$$

$$= -2\, e_2(k)\, d(k-i) \; ; \quad i = 1, 2, \ldots, N \tag{19b}$$

Hence, the resultant updating equations are, with $\mu > 0$

$$\gamma_j(k+1) = \gamma_j(k) + 2\,\mu\, e_2(k)\, u(k-j) \; ; \quad j = 0, 1, \ldots, M \tag{20.a}$$

$$\delta_i(k+1) = \delta_i(k) + 2\,\mu\, e_2(k)\, d(k-i) \; ; \quad i = 1, 2, \ldots, N \tag{20.b}$$

such that

$$e_2(k) = d(k) - \underline{X}^T(k)\, \hat{\underline{\theta}}(k) \tag{21}$$

is minimized, with $\underline{X}(k)$ is as in (10) and

$$\hat{\underline{\theta}}(k) = [\gamma_0(k) \; . \; . \; . \; \gamma_M(k) \mid \delta_1(k) \; \ldots \; \delta_N(k)\,]^T \tag{22}$$

defining the output of the auxiliary two input-one output adaptive FIR filter. Note that a $\delta_0 = 0$ coefficient could have been included in (13), such that for $N = 0$ $f(k)$ becomes a MA on $u(k)$ alone and (20) reduces to (20.a), which is the classical LMS algorithm. Hence this latter algorithm is an interpolation of (20), and we will refer to (20) as a extended LMS (ELMS) algorithm. The structure of an IIR adaptive filter using (20) to tune the coefficents of (13) such that MSE minimization of (17) is sought is shown in Fig. 1.

Next, we examine the convergence properties of the ELMS algorithm studying the characteristics of the indirect MSE minimization of (16) through the minimization of (17) as described above.

CONVERGENCE ANALYSIS FOR THE ELMS ALGORITHM

As mentioned before, even though the system identification is only a particular mode of operation for an adaptive filter, it is advantageous to analyze the behaviour of the filter under these conditions so that theoretical results can be drawn from the systems identification literature. In this spirit, for the convergence analysis of the ELMS algorithm, the approach to be followed here is to show that it can be cast into a more general equation error parameter estimation formulation [27], hence allowing the interpolation of the results obtained for this broader problem.

Equation error parameter estimation

The equation error formulation of parameter estimation is amply documented in [27]. The system under identification is assumed to be realistically modeled by the linear-in-the-parameters relation

$$y(k) = \underline{\theta}^T \underline{X}(k) = \underline{X}^T(k)\, \underline{\theta} \tag{23}$$

between the input information vector X(k) and the scalar output y(k), where $\underline{\theta}$ is a vector of parameters. This formulation has been extended to the multi-output case by Johnson [31]. The identifying model is of the form

$$y(k) = \underline{X}^T(k)\, \hat{\underline{\theta}}(k) \tag{24}$$

where $\hat{\underline{\theta}}(k)$ is the k^{th} approximation of $\underline{\theta}$.

Clearly, an ARMA process such as that of (1) can be cast into the form of (23) by defining

$$\underline{\theta} = [b_0\ b_1 \ \dots\ b_{Md} \mid a_1\ a_2 \ \dots\ a_{Nd}]^T \tag{25.a}$$

$$\underline{X}(k) = [u(k)\ u(k-1) \ \dots\ u(k-Md) \mid d(k-1)\ d(k-2) \ \dots\ d(k-Nd)]^T \tag{25.b}$$

then $$d(k) = \underline{X}^T(k)\, \underline{\theta} \tag{26}$$

and the identifier would be

$$\hat{d}(k) = \underline{X}^T(k)\, \hat{\underline{\theta}}(k) \tag{27}$$

It is clear that (27) and the adaptive auxiliary filter used to computer $e_2(k)$ in (21), are of the same form if $M = M_d$ and $N = N_d$ for $\hat{\underline{\theta}}(k)$ in (22). Given enough complexity for this auxiliary filter, and since (25.a) can be augmented with (M - Md) MA and (N - Nd) AR zero coefficients, we may assume M = Md and N = Nd without loss of generality.

Following the notation of (11), $y_a(k)$ instead of d(k) will be used subsequently for the identifier of (27). Thus, we have that

$$\begin{aligned} e_2(k) &= d(k) - y_a(k) \\ &= \underline{X}^T(k)\, \underline{\theta} - \underline{X}^T(k)\, \hat{\underline{\theta}}(k) \\ &= \underline{X}^T(k)\, \tilde{\underline{\theta}}(k) \end{aligned} \tag{28}$$

where $\tilde{\underline{\theta}}(k) = \underline{\theta} - \hat{\underline{\theta}}(k)$ (29)

Perfect Measurements case. If the input and the desired signal can be properly assumed to be measurable without any error, i.e., they are noise-free, then in [27] the following gradient descent algorithm is proposed to update $\hat{\underline{\theta}}(k)$ such that (28) is minimized

$$\hat{\underline{\theta}}(k+1) = \hat{\underline{\theta}}(k) + 2\,\underline{R}(k)\,e_2(k)\,\underline{X}(k) \tag{30}$$

where $\underline{R}(k)$ is a $(M + N + 1)$ X $(M + N + 1)$ symmetric weighting matrix.

By using (28), and subtracting (30) from (25.a), the following identification-error system is obtained.

$$\tilde{\underline{\theta}}(k+1) = [\underline{I} - 2\,\underline{R}(k)\,\underline{X}(k)\,\underline{X}^T(k)]\,\tilde{\underline{\theta}}(k) \tag{31}$$

where $\underline{I}$ denotes the identity matrix. For (31) above, it can be verified that $\tilde{\underline{\theta}}_e = \underline{0}$ is the only possible equilibrium point, for a general input, and thus a candidate for a potentially asymptotically stable point in the large [27].

Choosing the weighting matrix $\underline{R}(k)$ to be

$$\underline{R}(k) = \mu\,\underline{I}\,, \quad \mu \in \mathbb{R}, \quad \mu > 0 \tag{32}$$

then (30) can be decomposed into the form of the ELMS algorithm of (20). By constructing a scalar function $V(\tilde{\underline{\theta}},k)$ given by

$$\underline{V}(\tilde{\underline{\theta}},\, k) = \tilde{\underline{\theta}}^T(k)\,\tilde{\underline{\theta}}(k) \tag{33}$$

it is shown in [27] that this function will be a Lyapunov function for the free system of (31) if

$$0 < \mu < \frac{1}{\underline{X}^T(k)\underline{X}(k)}\,, \quad \forall\, k \in \mathbb{N} \cup \{0\} \tag{34}$$

Then, by the Lyapunov Stability Theorem [27], we will almost always have the following convergence condition

$$\hat{\underline{\theta}}(k) \longrightarrow \underline{\theta}, \qquad k \longrightarrow \infty \tag{35}$$

regardless of the initial estimate $\hat{\underline{\theta}}(0)$. This condition is almost always true because if $\tilde{\underline{\theta}}(k)$ and $\underline{X}(k)$ are orthogonal, then $e_2(k)$ in (28) will be zero when $\tilde{\underline{\theta}}(k) \neq \underline{0}$. To avoid this situation $\underline{X}(k)$ must be sufficiently rich [26], [27].

Clearly, if $M \geq M_d$ and $N \geq N_d$ for the adaptive version of (13) depicted in Fig. 1, then given (35), $e_2(k) \longrightarrow 0$ asymptotically, and from (16) and (17), $f(k) \longrightarrow d(k)$ asymptotically as was our ultimate goal.

Noisy measurements case. The problem of identifying $\underline{\theta}$ in (26) becomes a stochastic estimation task if $d(k)$ and $u(k)$ can only be measured in the presence of additive noise, that is, if the measured signals are of the form

$$d_m(k) = d(k) + v(k) \tag{36}$$

$$u_m(k) = u(k) + n(k) \tag{37}$$

where v(k) and n(k) are independent, zero-mean random sequences. Define the measured information vector as

$$\underline{X}m(k) = [u_m(k)\ u_m(k-1)\ \ldots\ u_m(k-M) \mid d_m(k-1)\ d_m(k-2)\ \ldots\ d_m(k-N)]^T \tag{38}$$

The equation above can be rewritten as

$$\underline{X}_m(k) = \underline{X}(k) + \underline{N}(k) \tag{39}$$

with $\underline{X}(k)$ being the perfect measurements information vector and

$$\underline{N}(k) = [n(k)\ n(k-1)\ \ldots\ n(k-M) \mid v(k-1)\ v(k-2)\ \ldots\ v(k-N)]^T \tag{40}$$

then the identifier of (27) becomes

$$y_{a,m}(k) = \underline{X}_m(k)\ \hat{\underline{\theta}}(k) \tag{41}$$

and

$$e_{2,m}(k) = d(k) - \underline{X}_m\ \hat{\underline{\theta}}(k) \tag{42}$$

Following the developments in [27], define

$$E[\underline{N}(k)\ \underline{N}^T(k)] = \underline{\Sigma}_n \tag{43}$$

$$E[\underline{X}(k)\ \underline{X}^T(k)] = \underline{\Omega} \tag{44}$$

If the following conditions are met

$$E[\underline{X}(k)\ \underline{N}^T(k)] = \underline{0} \tag{45}$$

$$E[v(k)\ \underline{N}(k)] = \underline{0} \tag{46}$$

$$E[\underline{X}(k)\ \underline{X}^T(k) \mid \hat{\underline{\theta}}(k)] = \underline{\Omega} \tag{47}$$

where (47) is a conditional expectation, and if $\hat{\underline{\theta}}(k)$ in (22) is updated by means of the following algorithm

$$\hat{\underline{\theta}}(k+p) = \hat{\underline{\theta}}(k) + 2\mu\ e_{2,m}(k)\ \underline{X}_m(k) \tag{48}$$

or equivalently

$$\gamma_j(k+p) = \gamma_j(k) + 2\ \mu\ e_{2,m}(k)\ u_m(k-j), \quad 0 \leq j \leq M \tag{49.a}$$

$$\delta_i(k+p) = \delta_i(k) + 2\ \mu\ e_{2,m}(k)\ d_m(k-i), \quad 0 < i \leq N \tag{49.b}$$

then the mean convergent parameter estimates, if they exist, are

$$\lim_{k\to\infty} \{E[\hat{\underline{\theta}}(k)]\} \triangleq \hat{\underline{\theta}}(\infty) = (\underline{\Omega} + \underline{\Sigma}_n)^{-1}\ \underline{\Omega}\ \underline{\theta} \tag{50}$$

where the value of p in (48) is chosen to uncorrelate $[\ \underline{X}(k)\ \underline{X}^T(k)\ ]$ with $\hat{\underline{\theta}}(k)$, so that (47) is satisfied.

Restricting to the commonly assumed situation where the input u(k) is measured perfectly, and the desired signal is measured in the presence of zero-mean white noise, then (40) becomes

$$\underline{N}(k) = [0\ 0\ \ldots\ 0 \mid v(k-1)\ v(k-2)\ \ldots\ v(k-N)]^T \tag{51}$$

where v(k) is now a zero-mean white random sequence. Clearly (45) and (46) are satisfied, and

$$\underline{\Sigma}_n = \left[\begin{array}{c|c} \underline{0} & \underline{0} \\ \hline \underline{0} & \underline{\Sigma}_v \end{array}\right] \tag{52}$$

where $\underline{\Sigma}_v = \text{diag}\ \{\sigma^2\}$,

is the autocorrelation matrix of $\underline{V} = [v(k-1)\ v(k-2)\ \ldots\ v(k-N)]^T$.

Let's examine the condition in (47). First note that $\hat{\underline{\theta}}(k)$ depends on $\underline{X}_M(k-p)$ through (48), and due to the IIR characteristics of (1), d(k) will depend on all previous values d(k-1), d(k-2), ..., d(0). Thus even assuming that an appropriate choice of p in (48) uncorrelates the part due to u(·) in $\underline{X}(k)$ and $\underline{X}(k-p)$, this will not be possible for the part due to d(·). However, if the system is to be stable, the effects of the value of the output at any given instant in time over future values must die out asymptotically. Otherwise the output would diverge as $k \rightarrow \infty$. Hence, it is possible to choose p in (48) such that (47) is satisfied within a given range of accuracy. If $p \rightarrow \infty$, (47) is satisfied. By choosing a finite p, (50) would constitute an approximate formula for the biasing in $\hat{\underline{\theta}}(\infty)$.

The inportant conclusion from the above comment is that whenever there is noise present in the measurement of d(k) and u(k), the estimates for $\underline{\theta}$ obtained through (48) will be biased. Thus the adaptive filter depicted in Fig. 1 will yield a suboptimal minimization of (16) when updated using (49).

Recall that it was said that the mean of the estimates for $\underline{\theta}$ obtained by use of (48), given (45) - (47) are met, would be given by (50) if there exists a steady-state value for $\hat{\underline{\theta}}(k)$. The proof of the existence of such a value is not a trivial task, and in [27] conditions are given for the case when u is replaced by a vanishing step-size. However, preliminary simulations show the convergence of (48) for constant , at least for relatively high signal to noise ratios. Further studies on this issue are being undertaken.

SUMMARY

We have shown that the ELMS algorithm can yield global minimization of (16), given three conditions: (i) the desired signal is generated by an ARMA process as described by (1), (ii) enough complexity is provided for the adaptive filter to span all the modes of (1), and (iii) there is no noise present in the measurements. If one or both of the first two conditions are not met, then $\hat{\underline{\theta}}(k)$ will converge to (12), minimizing globally (17) but not (16). If only the last condition is not met, then $\hat{\underline{\theta}}(\infty)$ will constitute a biased estimate of (25.a). The last two situations result in a suboptimal performance of the adaptive recursive filter. However, the low computational complexity and the simple conditions required for the ELMS algorihtm makes it appealing as a substitute for the LMS algorithm in those applications where the latter has proved of use, but a large number of tap weights are necessary.

REFERENCES

[1] B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," 1960 IRE WESCON Convention Record, pt. 4, pp. 96-104.

[2] B. Widrow, "Adaptive Filters," in Aspects of Network and System Theory, R. E. Kalman and N. Declaris, Eds., New York: Holt, Rinehart, and Winston, 1971, pp. 563-587.

[3] B. Widrow, J. M. McCool, and M. Ball, "The Complex LMS Algorithm," Proceedings of the IEEE, Vol. 63, no. 4, pp. 719-720, April 1975.

[4] B. Widrow, J. M. McCool, M.G. Larimore and C.R. Johnson, Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," Proceedings of the IEEE, Vol 64, no. 8, pp. 1151, 1162, Aug. 1976.

[5] E. Walach and B. Widrow, "The Least Mean Fourth (LMF) Adaptive Algorithm and Its Family," IEEE Transactions on Information Theory, Vol. IT-30, No. 2, pp. 275-283, March 1984.

[6] T. Aprille, "Filtering and Equalization for Digital Transmission," IEEE Communications Magazine, pp. 17-24, March 1983.

[7] J.G. Proakis, Digital Communications, New York: McGraw-Hill, 1983.

[8] B. Widrow and E. Walach, "Adaptive Signal Processing for Adaptive Control," Proceedings of the 1984 IEEE International Conference on Acoustic, Speech, Signal Processing, San Diego, CA, March 1984, pp. 21.1.1-21.1.4.

[9] C.R. Johnson, Jr. "Input Matching, Error Augmentation, Self-Tuning and Output Error Identification: Algorithmic Similarities in Discrete Model Following," IEEE Transactions on Automatic Control, Vol. AC-25, No. 4, pp. 697-703, August 1980.

[10] S.A. White, "An Adaptive Recursive Digital Filter," Ninth Annual Asilomar Conference on Circuits, Systems and Computers Conference Record, Pacific Grove, CA, November 1975, pp. 21-25.

[11] S.D. Stearns, G.R. Elliot and N. Ahmed, "On Adaptive Recursive Filtering," Tenth Annual Asilomar Conference on Circuits, Systems and Computers Conference Record, San Francisco, CA, November 1976, pp. 5-11.

[12] J.A. Cadzow, "Recursive Digital Filter Synthesis Via Gradient Based Algorithms," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-24, no. 5, pp. 349-355, October 1976.

[13] P.L. Feintuch, "An Adaptive Recursive LMS Filter," Proceedings of the IEEE, Vol. 64, no. 11, pp. 1622-1624, November 1976.

[14] C.R. Johnson, Jr., and M.G. Larimore, "Comments on and Additions to an Adaptive Recursive LMS Filter," Proceedings of the IEEE, Vol. 65, no. 9, pp. 1399-1402, September 1977.

[15] B. Widrow and J.M. McCool, "Comments on An Adaptive Recursive LMS Filter," Proceedings of the IEEE, Vol. 65, no. 9, pp. 1402-1404, September 1977.

[16] D. Parikh and N. Ahmed, "On an Adaptive Algorithm for IIR Filters," Proceedings of the IEEE, Vol. 66, no. 5, pp. 585-587, May 1978.

[17] S.D. Stearns, "Error Surfaces of Recursive Adaptive Filters," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-29, no. 3, pp. 763-766, June 1981.

[18] T.C. Hsia, "A Simplified Adaptive Recursive Filter Design," Proceedings of the IEEE, Vol. 69, No.9, pp. 1153-1155, September 1981.

[19] C.R. Johnson, Jr., "A Convergence Proof for a Hyperstable Adaptive Recursive Filter," IEEE Transaction on Informaton Theory, Vol. IT-25, no. 6, pp. 745-749, November 1979.

[20] J.R. Treichler, M.G. Larimore and C.R. Johnson, Jr., "On the Convergence Properties of the Simple Hyperstable Adaptive Recursive Filter (SHARF)," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Denver, Co, April 1980, pp. 997-1000.

[21] C.R. Johnson, Jr., "A Stable Family of Adaptive IIR Filters" Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Denver, CO, April 1980, pp. 1001-1004.

[22] M.G. Larimore, J.R. Treichler and C.R. Johnson, Jr., "SHARF: An Algorithm for Adapting IIR Digital Filters," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-28, no. 4, pp. 428-440, August 1980.

[23] C.R. Johnson, Jr., I.D. Landau, T. Taylor and L. Dugard, "On Adaptive IIR Filters and Parallel Adaptive Idenfifiers with Adaptive Error Filtering," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, GA, March 1981, pp. 538-541.

[24] C.R. Johnson, Jr., M.G. Larimore, J.R. Treichler and B.D.O. Anderson, "SHARF Convergence Properties," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, no.3, pp. 659-670, June 1981.

[25] M.G. Larimore, "Hyperstability and Adaptive Filtering," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, no.2, pp. 319-320, April 1981.

[26] C.R. Johnson, Jr., "Adaptive IIR Filtering: Current Results and Open Issues," IEEE Transactions on Informatior Theory, Vol. IT-30, No. 2, pp. 237-250, March 1984.

[27] J.M. Mendel, Discrete Techniques of Parameter Estimation: The Equation Error Formulation. New York: Marcel Dekker, 1973.

[28] T.K. Sarkar, S.A. Dianat, and D.D. Weiner, "A Discussion of Various Approaches to The Linear Systems Identification Problem," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-32, No. 3, pp. 654-656, June 1984.

[29] K. Steiglitz, "On the Simultaneous Estimation of Poles and Zeros in Speech Analysis," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-25, no. 3, pp. 229-234, June 1977.

[30] R.E. Kalman, "Design of Self-Optimizing Control Systems," Transactions of the ASME, vol. 80, pp. 468-478, Feb. 1958.

[31] C.R. Johnson, Jr., "Adaptive Parameter Matrix and Output vector Estimation via an Equation Error Formulation," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-9, no. 7, pp. 392-3297, July 1979.

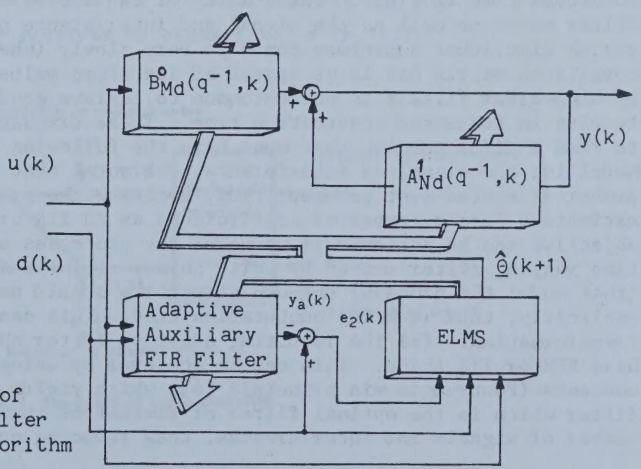


Fig. 1 Structure of an adaptive IIR filter using the ELMS algorithm

ADAPTIVE NOISE CANCELLING AND NONLINEAR ESTIMATION THEORY

AHMED ABUTALEB
Medical University of South Carolina
College of Graduate Studies, Department of Biometry

Abstract:

A new adaptive noise canceller is proposed. It is based on optimal control and nonlinear estimation theories. This approach was compared to the linear mean square (LMS) method, and its performance proved to be superior.

I. Introduction:

Noise cancelling is a variation of optimal filtering that is highly advantageous in many applications. It makes use of an auxiliary or reference input derived from one or more sensors located at points in the noise field where the signal is weak or undetectable. This input is filtered and subtracted from a primary input containing both signal and noise. As a result the primary noise is attenuated or eliminated by cancellation.

In typical implementation of adaptive noise cancelling (ANC) the filtering is achieved using a finite impulse response (FIR) filter. The filter coefficients are then adjusted using a gradient search algorithm such as the linear mean square (LMS) [1]. Another approach [2,3] is to model the interference (reference input) as the output of infinite impulse response (IIR) filter (with unknown coefficients) driven by white noise. The IIR filter coefficients are adjusted using, e.g., recursive prediction error algorithm [4]. Thus, an estimate of the interference signal is obtained and subtracted from the primary input to yield an estimate of the signal. This process is equivalent to passing the primary input through an IIR filter to obtain the signal. These implementations of the ANC perform well in many situations. Their performance is, however, suboptimal due to the following: (1) The FIR or IIR filters structure (both are linear time invariant) are only an approximation of the optimal filter (which may be nonlinear time varying). The quality of the approximation depends on the filter order as well as the signal and interference properties. (2) The search algorithms sometimes converge very slowly (when the interference covariance matrix has large spread of its eigen values [5]. (3) The use of high-order filters is not uncommon to achieve good performance which results in increased computation time. These drawbacks were the motivation to find a different ANC that must have the following properties: (a) In modelling the signal or interference, the model that uses the minimum number of states must be used, thus, avoiding the problem of having to estimate a larger number of coefficients as in IIR or FIR [2]. This objective can be achieved if we model the processes as output of a linear time varying filter driven by white noise. (b) The time varying filters (that model the process) unknown structures should not be estimated explicitly, thus reducing computation time. This can be achieved through transformation. (c) The resulting adaptive filter should not be assumed to have FIR or IIR shape. This can be achieved by using optimal control concepts (Pontryagin min principle [6]) which yields a nonlinear adaptive filter which is the optimal filter of dimensions that are twice the total number of signals and interferences, thus reducing computation time.

This paper presents a new approach for cancelling interference from signal measurements that posseses the above mentioned properties. In the proposed method, both the signal and the interference are modeled as nonstationary processes. Thus, the signal or interference can be represented by the equation $x(t) = \alpha(t)\ x\ (t) + w(t)$, where $x(t)$ is the signal (or interference), $w(t)$ is white noise with level 1, and $\alpha(t)$ is an unknown time function to be estimated. The above equation models the signal (or interference) as the output of a linear time varying filter driven by white noise. Pontryagin minimum principle [6] was used, in a novel way, to derive a set of nonlinear equations that relate the interference and signal estimates to the noisy observation. This approach resulted in a two point boundary value problem which can not be solved in real time. Using the method of invariant imbedding [7], the two point boundary value problem was transformed into one point boundary value problem, and the resulted equations were solved in real time. The solution of these equations yielded the estimates of the signal and interference.

The proposed method was compared to the linear mean square (LMS) algorithm of Widrow [1], where it was shown to have better performance (as measured by the error between the real signal and its estimated value). Four examples are given: (1) Interference and signal are both sinusoidal waves but at two different frequencies. (2) Interference is sinusoidal wave while the signal is a Guass-Markov process. (3) Interference is a Guass-Markov process while the signal is simusoidal. (4) Both are Guass-Markov processes. The effects of signal to noise ratios (SNR) and signal to interference ratios (SIR) were also studied for these cases and compared to the LMS performance. In all cases, the proposed method showed better performance. The computer time needed to arrive at the estimates, using the proposed method, was around 20% of the time for the LMS algorithm, which is an advantage of the method. In Section II, the proposed Pontryagin ANC is introduced. In Section III, simulation results are shown for both the LMS algorithm and the Pontryagin ANC. Finally, in Section IV, conclusion and discussion are presented.

II. ANC using Pontryagin minimum principle:

A block diagram of the adaptive noise cancelling problem is shown in Figure 1. The objective is to filter the signal $[x_2(t)]$ from the interference $[x_1(t)]$ and receiver or channel noise $[n_2(t)]$.

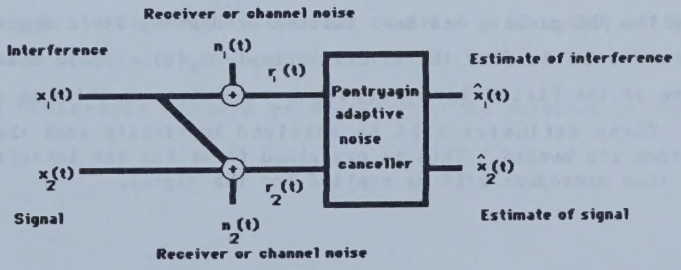


Fig. 1. The adaptive noise canceller.

(a) **Linear time varying filter:**

Let the interference and signal be the ouput of linear time varying filter that is described by the followng differential equation

$$\dot{x}_1(t) = \alpha_1(t)\, x_1(t) + w_1(t) \qquad (1)$$

and

$$\dot{x}_2(t) = \alpha_2(t)\, x_2(t) + w_2(t) \qquad (2)$$

where

$x_1(t)$ = interference at time t

$x_2(t)$ = signal at time t

'$\cdot$' = derivative with respect to time

$w_1(t)$ = white Gaussian noise with energy level 1

$w_2(t)$ = white Gaussian noise with energy level 1 independent of $w_1(t)$

$\alpha_1(t)$, $\alpha_2(t)$ = unknown time functions that represent the time varying filter.

The observation available at the receiver is

$$r_1(t) = x_1(t) + n_1(t) \qquad (3)$$

and

$$r_2(t) = x_2(t) + x_1(t) + n_2(t) \qquad (4)$$

where

$r_1(t)$ = reference input at time t

$r_2(t)$ = primary input at time t

$n_1(t)$ = observation noise of the interference sensor and channel that is independent of $x_1(t)$, $x_2(t)$, $w_1(t)$ $w_2(t)$ and $n_1(t)$. The energy level is dependent upon the interference to noise ratio of the channel.

$n_2(t)$ = observation noise of the primary receiver and channel, it is independent of $x_1(t)$, $x_2(t)$, $w_1(t)$, $w_2(t)$ and $n_1(t)$. The energy level is determined by the signal to noise ratio of the channel.

Thus the ANC problem has been reformulated using State space analysis. The objective is to find the signal estimate $x_2(t)$. It is clear that the estimation of the filter time varying parameters, $\alpha_1(t)$ and $\alpha_2(t)$, are needed. These estimates will be obtained implicitly such that no extra calculations are needed. This is explained first for the interference and then the same procedure will be applied for the signal.

(b) **Estimation of the interference:**

The objective is to estimate the interference, $x_1(t)$, that is governed by the equation

$$\dot{x}_1(t) = \alpha_1(t)\, x_1(t) + w_1(t) \tag{1}$$

where $\alpha_1(t)$ is unknown, and using only the observation

$$r_1(t) = x_1(t) + n_1(t) \tag{3}$$

The approach presented here is based on Pontryagin minimum principle. The basic idea is to interpret $\alpha_1(t)$ and $w_1(t)$ as unknown controllers that try to drive the system state $x_1(t)$, to follow (or track) a trajectory that is given by the noisy observation $r_1(t)$ of equation (3). Thus, the stochastic estimation problem has been transformed into a deterministic optimization problem, with dynamic constraints, that yields itself easily to Pontryagin minimum principle. Before we proceed further, the reader should realize that there is a duality between estimation and optimization as explained by many researchers. For example, Kalman filter was derived using Pontryagin minimum principle [8].

We now turn our attention to the optimization problem at hand. We first define a cost function that is actually the square error:

$$\text{Let} \quad J = \frac{1}{2} \int_0^T d\delta \, \{ [r_1(\delta) - x_1(\delta)]^2 + \mu_{\alpha_1} \alpha_1^2(\delta) + \mu_{w_1} w_1^2(\delta) \} \tag{5}$$

where T is the running current time, μ_{α_1} and μ_{w_1} are weights that represent the a priori knowledge about the range of variations of $\alpha_1(t)$ and $w_1(t)$ respectively. Imbedded in the weights is the interference to noise ratio, i.e. as the interference to noise ratio increases μ_{α_1} and μ_{w_1} decrease. They affect speed of convergence, stability of the equations, and accuracy of the estimation procedures. Their values are chosen after running some preliminary experiments with the situation at hand.

The problem now can be stated as follows: Find the values of $\alpha_1(t)$, $w_1(t)$ and $x_1(t)$ that minimize the cost function J of equation (5) subject to the dynamic constraints of equation (4).

Using Pontryagin minimum principle [6], the solution is obtained as follows:

(i) Form Pontryagin H function.

$$H(x_1, r_1, \alpha_1, w_1, \lambda; t) = \frac{1}{2}[r_1(t) - x_1(t)]^2 + \frac{1}{2}\mu_{\alpha_1}\alpha_1^2(t) + \frac{1}{2}\mu_{w_1}w_1^2(t) + \lambda(t)[\alpha_1(t)x_1(t) + w_1(t)] \quad (6)$$

where $\lambda(t)$ is known as the lagrange multiplier or the costate.

(ii) Minimize $H(x_1, r_1, \alpha_1, w_1, \lambda; t)$ with respect to the unknown controllers; $\alpha_1(t)$ and $w_1(t)$:

$$\frac{\partial H(x_1, r_1, \alpha_1, w_1, \lambda; t)}{\partial \alpha_1(t)} = 0 = \mu_{\alpha_1}\alpha_1(t) + \lambda(t)x_1(t) \quad (7)$$

which yields

$$\alpha_1(t) = -\lambda(t)x_1(t)/\mu_{\alpha_1} \quad (8)$$

and for $w_1(t)$ we get

$$\frac{\partial H(x_1, r_1, \alpha_1, w_1, \lambda; t)}{\partial w_1(t)} = 0 = \mu_{w_1}w_1(t) + \lambda(t) \quad (9)$$

which yields

$$w_1(t) = -\lambda(t)/\mu_{w_1} \quad (10)$$

(iii) Substitute the values obtained for $\alpha_1(t)$ and $w_1(t)$ to obtain the minimum H function (H^*):

$$H^*(x_1, r_1, \lambda; t) = \frac{1}{2}[r_1(t) - x_1(t)]^2 + \frac{1}{2}\mu_{\alpha_1}\frac{\lambda^2(t)x_1^2(t)}{\mu_{\alpha_1}^2} + \frac{1}{2}\mu_{w_1}\frac{\lambda^2(t)}{\mu_{w_1}^2} + \lambda(t)\left[-\frac{\lambda(t)x_1^2(t)}{\mu_{\alpha_1}} - \frac{\lambda(t)}{\mu_{w_1}}\right] \quad (11)$$

which is reduced to

$$H^*(x_1, r_1, \lambda; t) = \frac{1}{2}[r_1(t) - x_1(t)]^2 - \frac{\lambda^2(t)}{2}\left[\frac{x_1^2(t)}{\mu_{\alpha_1}} + \frac{1}{\mu_{w_1}}\right] \quad (12)$$

(iv) Solve the following 2 differential Equations:

$$\dot{x}_1(t) = \frac{\partial H^*(x_1, r_1, \lambda; t)}{\partial \lambda}$$

$$= - \lambda(t) \left[\frac{x_1^2(t)}{\mu_{\alpha_1}} + \frac{1}{\mu_{w_1}} \right] \tag{13}$$

and

$$\dot{\lambda}(t) = - \frac{\partial H^*(x_1, r_1, \lambda; t)}{\partial x_1(t)}$$

$$= [r_1(t) - x_1(t)] + \frac{\lambda^2(t)\, x_1(t)}{\mu_{\alpha_1}} \tag{14}$$

The initial condition $x_1(0)$ is assumed to be known with some uncertainty. The other initial condition $\lambda(0)$, however, is unknown. The final condition $\lambda(T)$ is known and equals 0, ($\lambda(T)=0$, because $x_1(T)$ is unknown), this result comes from the derivation of the minimum principle [8].

Thus, we have two point boundary value problem that can not be solved in real time. The solution of this problem will yield the desired estimate of $x_1(t)$. The method of invariant imbedding [7] is used to transform the two point boundary value problem to one point boundary value problem that can be solved in real time.

(v) Invariant imbedding:

Rewrite equations (13) and (14) in a compact form as follows:

$$\dot{x}_1(t) = \gamma(x_1, \lambda; t) \tag{15}$$

and

$$\dot{\lambda}(t) = \eta(x_1, \lambda; t) \tag{16}$$

where

$$\gamma(x_1, \lambda; t) = - \lambda(t) \left[\frac{x_1^2(t)}{\mu_{\alpha_1}} + \frac{1}{\mu_{w_1}} \right] \tag{17}$$

and

$$\eta(x_1, \lambda; t) = [r_1(t) - x_1(t)] + \frac{\lambda^2(t)\, x_1(t)}{\mu_{\alpha_1}} \tag{18}$$

The method of invariant imbedding yields the following differential equations with known initial conditions.

$$\dot{\hat{x}}_1(t) = \gamma(\hat{x}_1;t) + P(t)\,\eta(\hat{x}_1;t) \tag{19}$$

and

$$\dot{P}(t) = \frac{\partial\gamma}{\partial\hat{x}_1} P(t) - P(t)\frac{\partial\eta}{\partial\lambda} + P(t)\,\frac{\partial\eta}{\partial\hat{x}_1} P(t) - \frac{\partial\gamma}{\partial\lambda} \tag{20}$$

where

$\hat{x}_1(t)$ is the estimate of $x_1(t)$

$$\gamma(\hat{x}_1;t) = \gamma(x_1, \lambda;t)\Big|_{\lambda=0} \tag{21}$$

$$\eta(\hat{x}_1;t) = \eta(x_1,\lambda;t)\Big|_{\lambda=0} \tag{22}$$

$$\frac{\partial\gamma}{\partial\hat{x}_1} = \frac{\partial\gamma(x_1, \lambda;t)}{\partial x_1}\Big|_{\lambda=0} \tag{23}$$

$$\frac{\partial\eta}{\partial\lambda} = \frac{\partial\eta(x_1, \lambda;t)}{\partial\lambda}\Big|_{\lambda=0} \tag{24}$$

$$\frac{\partial\eta}{\partial\hat{x}_1} = \frac{\partial\eta(x_1, \lambda;t)}{\partial x_1}\Big|_{\lambda=0} \tag{25}$$

$$\frac{\partial\gamma}{\partial\lambda} = \frac{\partial\gamma(x_1, \lambda;t)}{\partial\lambda}\Big|_{\lambda=0} \tag{26}$$

Substitute for the different values in equations (21) - (26) into equations (19) and (20), we arrive at:

$$\dot{\hat{x}}_1(t) = P(t)\ [r_1(t) - \hat{x}_1(t)] \tag{27}$$

which is the state estimation equation

and

$$\dot{P}(t) = -P^2(t) + \frac{\hat{x}_1^2(t)}{\mu_{\alpha_1}} + \frac{1}{\mu_{w_1}} \tag{28}$$

which is the error variance equation with the known initial conditions

$$\hat{x}_1(0) = x_1(0) \text{ and } P(0) = P_0$$

Notice that P_0 represents the error variance and it is the uncertainty in the initial condition $x_1(0)$ i.e., if the uncertainty is low P_0 is small. Since the observation time is in the order of seconds, (which may amount to thousands of samples) the choice of $x_1(0)$ or P_0 does not affect much the estimates. It should be realized that the steps and equations presented in this section constitute a new nonlinear filter.

(c) **Estimation of the signal:**

Since the interference $[x_1(t)]$ has been estimated independently (eqns. 19-20), its estimated value is subtracted from $r_2(t)$ to yield the noisy observation $[r_2(t) - \hat{x}_1(t)]$ of the signal $x_2(t)$. Thus, the same procedure used with the interference is now used to estimate the signal.

III. Simulation and results:

Four cases were simulated for different values of the signal and interference and at different signal to noise ratio (SNR) and interference to noise ratio (INR). In all cases, the LMS algorithm (with 5 weights) is used and its results are compared to that of Pontryagin ANC. The performance is measured by the root mean square (RMS) value of the error between the true and estimated values of the signal. Two waveforms were used for both signal and interference; two sinusoids, $x_1(t) = 2 \sin 2\pi f_1 t$, and $x_2(t) = 2 \sin 2\pi f_2 t$, where $f_1 = 20$ HZ and $f_2 = 10$ HZ. Two Gauss-Markov processes, each is the output of a different low pass filter driven by white noise; $\dot{x}_1(t) = \frac{1}{a}[x_1(t) + v_1(t)]$ and $\dot{x}_2(t) = \frac{1}{b}[x_2(t) + v_2(t)]$, where $a = 1$, $b = 0.5$, v_1 and v_2 are two independent zero mean gaussian white noise with energy level 1. Four combinations of the signal and interference are used and the performance is shown in Figure 2 - Figure 5. It is clear that as SNR or INR decreases, the Pontryagin ANC shows increasingly better performance than LMS ANC.

IV. Conclusion and Discussion

In this paper, a new approach for adaptive noise cancelling was introduced. The improvement in performance and speed were mainly due to introducing a new nonlinear filter that is based on Pontryagin minimum principle. This filter performance was compared to that of the conventional LMS algorithm for different simulation cases. The results are in favor of the new scheme. Several issues remain to be studied concerning the stability and convergence of the proposed nonlinear filter under different conditions.

REFERENCES

(1) B. Widrow, J. R. Glover, Jr., J. M. McCool, J. Kauntiz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong Jr., and R. C. Goodlin, ''Adaptive Noise Cancelling: Principles and Applications'', Proceedings of the IEEE, Vol. 63, pp. 1692-1716, Dec. 1975,.

(2) B. Friedlander, ''System Identification Techniques for Adaptive Noise Cancelling'', IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-30, #5, pp. 699-708, Oct., 1982.

(3) P. Heinonen, T. Saramaki, J. Malmivuo, and Y. Neuvo, ''Periodic Interference Rejection Using Coherent Sampling and Waveform Estimation'', IEEE Trans. Circuits and Systems, Vol. CAS-31, #5, PP. 438-446, MAY, 1984.
(4) L. Ljung and T. Soderstrom, ''Theory and Practice of Recursive Identification'', Cambridge, MA: M.I.T. Press 1982.
(5) S. S. Narayan, A. M. Peterson, and M. J. Narasimha, ''Transform Domain LMS Algorithm'', IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-31, #3, pp. 609-615, June, 1983.
(6) L. S. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mischenko, ''The Mathematical Theory of Optimal Processes'', Interscience Publishers, New York, 1962.
(7) R. E. Belman, H. H. Kagiwada, R. E. Kalaba, and R. Sridhar,'' Invariant Imbedding and Nonlinear Filtering Theory'', The Journal of the Astronautical Sciences. Vol. XIII, #3, pp. 110-115, May-June, 1966.
(8) D. G. Schultz and J. L. Melsa, ''State functions and linear control systems'', McGraw-Hill, New York 1967.

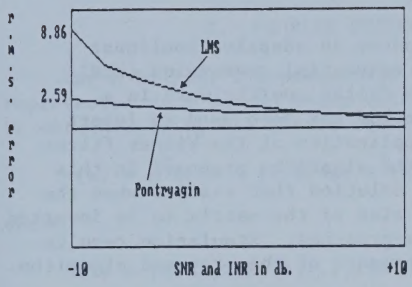


Fig. 2. Adaptive noise cancelling using LMS algorithm and Pontryagin filter, SNR=INR. Root mean square (r.m.s.) error is plotted versus SNR. Signal is sinusoid and interference is sinusoid with different frequency.

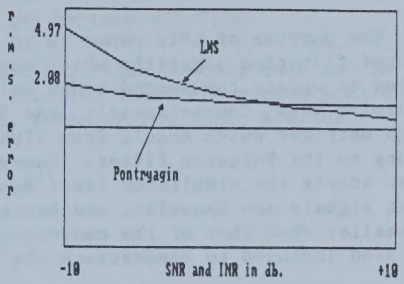


Fig. 3. Adaptive noise cancelling using LMS algorithm and Pontryagin filter, SNR=INR. Root mean square (r.m.s.) error is plotted versus SNR. Signal is sinusoid and interference is Gauss-Markov process.

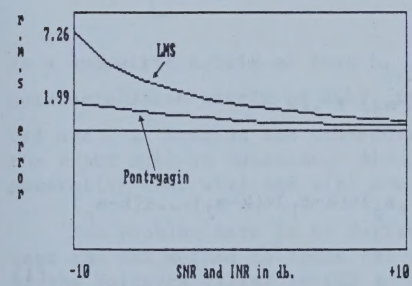


Fig. 4. Adaptive noise cancelling using LMS algorithm and Pontryagin filter, SNR=INR. Root mean square (r.m.s.) error is plotted versus SNR. Signal is Gauss-Markov process and interference is sinusoid.

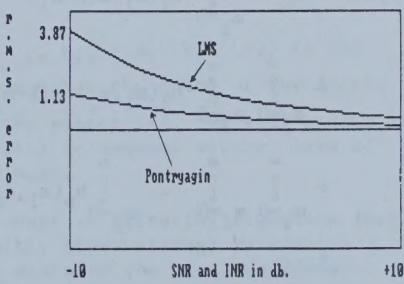


Fig. 5. Adaptive noise cancelling using LMS algorithm and Pontryagin filter, SNR=INR. Root mean square (r.m.s.) error is plotted versus SNR. Signal is Gauss-Markov process and interference is Gauss-Markov process.

ADAPTIVE NONLINEAR DIGITAL FILTER WITH SEQUENTIAL REGRESSION ALGORITHM

D.H. YOUN, K.K. YU and V.J. MATHEWS
Department of Electrical and Computer Engineering
The University of Iowa
Iowa City, Iowa 52242

ABSTRACT

The purpose of this paper is to introduce an adaptive nonlinear digital filtering algorithm which use the sequential regression (SER) method to update the second order Volterra filter coefficients in a recursive way. Conventionally, the SER method has been used to invert large matrices which result from direct application of the Wiener filter theory to the Volterra filter. However, the algorithm proposed in this paper adopts the simplified least squares solution that results when the input signals are Gaussian, and hence the size of the matrix to be inverted is smaller than that of the conventional approaches. Simulation results are also included to demonstrate the performance of the proposed algorithm.

1. INTRODUCTION

The Volterra series representation of nonlinear systems is an extension of the linear system theory. The output y(k) of a (causal) discrete-time, nonlinear system can be represented as a function of the input x(k) using the Volterra series as

$$\begin{aligned} y(k) = h_0 &+ \sum_{m_1=0}^{\infty} h_1(m_1)x(k-m_1) \\ &+ \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty} h_2(m_1,m_2)\, x(k-m_1)x(k-m_2) + \cdots \\ &+ \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty}\cdots\sum_{m_p=0}^{\infty} h_p(m_1,m_2,\ldots,m_p)x(k-m_1)x(k-m_2)\cdots x(k-m_p) \\ &+ \cdots \end{aligned} \tag{1}$$

where $h_p(m_1,m_2,\ldots,m_p)$ is the p-th order Volterra kernal [1] of the nonlinear system. This paper is concerned with adaptive identification of nonlinear systems that can be represented using a second order Volterra series. A least square (LS) solution is obtained for the system identification problem assuming Gaussian reference inputs and an adaptive filtering algorithm which computes the LS solution in a recursive manner using the sequential regression (SER) method is derived.

Conventionally, the SER method has been used to invert a large matrix which results from direct application of the Wiener filter theory to nonlinear systems [6]. However, the algorithm proposed in this paper adopts the simplified solution [5] that results when the input signals are Gaussian, and therefore the size of the matrix to be inverted is smaller than that of the conventional approach.

The rest of the paper is organized as follows: In Section 2, a formal statement of the problem is presented. A LS solution to the problem is derived in this section. The adaptive Volterra filter with SER algorithm is derived in Section 3. The effectiveness of the proposed algorithm is demonstrated using a simulation example in Section 4. Finally, we make the concluding remarks in Section 5.

2. PROBLEM STATEMENT AND OPTIMUM SOLUTION

Let H in Fig. 1 represent an unknown nonlinear system, that can be represented as a second order Volterra filter. Then, the output y(k) can be obtained using the matrix equation in terms of the input x(k) as

$$y(k) = A^T X(k) + tr\{B[X(k)X^T(k) - R_{xx}]\} \tag{2}$$

where

$$X(k) = [x(k), x(k-1), \ldots, x(k-N+1)]^T,$$

$$A = [a_0, a_1, \ldots, a_{N-1}]^T,$$

$$B = \begin{bmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,N-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,N-1} \\ \vdots & & & \\ b_{N-1,0} & b_{N-1,1} & \cdots & b_{N-1,N-1} \end{bmatrix}$$

is a symmetric matrix so that $b_{i,j} = b_{j,i}$, $R_{xx} = E\{X(k)X^T(k)\}$ is the autocorrelation matrix of X(k), tr{•} denotes the trace of the matrix {•} and $(\cdot)^T$ denotes the transpose of the vector (•). We will assume that the input x(k) is Gaussian. Also, it will be assumed without loss of generality that x(k) and y(k) are zero mean.

The problem here is to derive an adaptive filtering algorithm that uses the SER method to track the (possibly time-varying) parameters A and B of the Volterra filter in (2) so as to minimize the cost functional

$$C(\hat{A}(k), \hat{B}(k)) = \sum_{i=0}^{k} q(i,k)\{y(i) - [\hat{A}(k)]^T X(i) - tr[\hat{B}(k)(X(i)X^T(i) - \hat{R}_{xx}(k))]\}^2 \tag{3}$$

where $\hat{}$ denotes an estimated quantity and q(i,k) is a weighting function for the squared estimation error $[y(i) - \hat{y}_k(i)]^2$ at time i, when the estimates $\hat{A}(k)$, $\hat{B}(k)$ and $\hat{R}_{xx}(k)$ at time k are used to obtain the estimate $\hat{y}_k(i)$ of y(i). For analytical tractability, we will assume that the autocorrelation matrix R_{xx} is estimated at time k as

$$\hat{R}_{xx}(k) = \sum_{i=0}^{k} q(i,k)X(k)X^T(k) \ . \tag{4}$$

Then, the cost functional that is to be minimized becomes

$$C\{\hat{A}(k),\hat{B}(k)\} = \sum_{i=0}^{k} \{q(i,k)\{y(i) - \hat{A}^T(k)X(i) - tr[\hat{B}(k)\{X(i)X^T(i) - \sum_{j=0}^{k} q(j,k)X(j)X^T(j)\}]\}^2\} . \quad (5)$$

We can minimize the above cost functional with respect to $\hat{A}(k)$ by setting the gradient of $C\{\hat{A}(k),\hat{B}(k)\}$ with respect to $\hat{A}(k)$ to zero, i.e., by setting $\nabla_{\hat{A}(k)} C\{\hat{A}(k),\hat{B}(k)\} = 0$. After some straightforward computations, we obtain

$$\hat{R}_{yx}(k) = \hat{R}_{xx}(k)\overset{0}{A}(k) + \sum_{m,n=0}^{N-1} \hat{b}_{m,n}(k) \left[\sum_{i=0}^{k} q(i,k)x(i-m)x(i-n)X(i)\right] + \left[\sum_{j=0}^{k} q(j,k)X^T(j)\hat{B}(k)X(j)\right]\left[\sum_{i=0}^{k} q(i,k)X(i)\right], \quad (6a)$$

where

$$\hat{R}_{yx}(k) = \sum_{i=0}^{k} q(i,k)y(i)X(i) \quad (6b)$$

and $\overset{0}{A}(k)$ yields the minimum $C\{\hat{A}(k),\hat{B}(k)\}$.

If the weighting function $q(i,k)$ for $0 \leq i \leq k$ represents the impulse response function of a lowpass filter with unit gain at zero frequency, i.e., $\sum_{i=0}^{k} q(i,k)=1$, we can see that $\sum_{i=0}^{k} q(i,k)x(i-j)$ and $\sum_{i=0}^{k} q(i,k)x(i-m)x(i-n)x(i-j)$ approximate the mean and third order moment of $x(k)$, respectively, which means that second and third terms of (6a) are approximately zero. Substitution of this in (6b) results in the simplified expression for $\overset{0}{A}(k)$, which is given by

$$\overset{0}{A}(k) = \hat{R}_{xx}^{-1}(k)\,\hat{R}_{yx}(k) . \quad (7)$$

To obtain the LS solution for $\hat{B}(k)$, we once again set the gradient of $C\{\hat{A}(k),\hat{B}(k)\}$ with respect to $\hat{B}(k)$ to zero and obtain

$$\overset{0}{B}(k) = \frac{1}{2}\hat{R}_{xx}^{-1}(k)T_{yx}(k)\hat{R}_{xx}^{-1}(k) \quad (8)$$

where

$$T_{yx}(k) = \sum_{i=0}^{k} q(i,k)y(i)X(i)X^T(i) . \quad (9)$$

The derivation is straightforward and can be found in [3]. It makes use of the fact that for a Gaussian signal $x(k)$,

$$E\{x(k-i)x(k-j)x(k-m)x(k-n)\} = r_{xx}(i-j)r_{xx}(m-n) + r_{xx}(i-m)r_{xx}(j-n) + r_{xx}(i-n)r_{xx}(j-m) \quad (10)$$

and the approximate relationship

$$\sum_{i=0}^{k} q(i,k)x(i-m)x(i-n) \ldots x(i-j) \approx E\{x(k-m)x(k-n) \ldots x(k-j)\}. \quad (11)$$

In (10), $r_{xx}(i-j) = E\{x(k-i)x(k-j)\}$.

It may be pointed out that the LS solutions given by (7) and (8) have the same form as those derived for the optimal minimum mean squared estimates for A(k) and B(k) in [4,5].

Also, for the general case, when the relevant signals are stationary, applying the Wiener filter theory yields the optimum solution [6]

$$\tilde{H} = \tilde{R}_{xx}^{-1}\tilde{R}_{yx} \quad (12)$$

where

$$\tilde{H} = [a_0, a_1, a_2, \ldots, a_{N-1}, b_{0,0}, b_{0,1}, \ldots, b_{N-1,N-1}]^T$$

$$\tilde{R}_{xx} = E\{\tilde{X}(k)\tilde{X}^T(k)\}$$

$$\tilde{R}_{yx} = E\{y(k)\tilde{X}(k)\}$$

and

$$\tilde{X}(k) = [X(k), x(k-1), \ldots, x(k-N+1), x^2(k)-r_{xx}(0), X(k)X(k-1)-r_{xx}(1), \ldots, x^2(k-N+1) - r_{xx}(0)]^T .$$

From (12) it can be seen that computing the optimal solution requires inverting an $(N+N^2)x(N+N^2)$ matrix, which may be computationally very difficult for large values of N. In comparison, the solutions derived in (7) and (8) require inverting only an NxN matrix. The savings in computations involved is quite evident from the above discussion.

In the next section, we will develop an exponentially weighted SER algorithm for recursively computing $\hat{A}(k)$ and $\hat{B}(k)$.

3. THE EXPONENTIALLY WEIGHTED SER ALGORITHM

The SER algorithm has been used to update the optimum linear [2] and nonlinear [6] filter coefficients in a recursive manner. In this section, we adopt the LS solution in (7) and (8) and apply the SER algorithm to update the Volterra filter weights that minimize the cost functional given by (5) with the weighting function q(i,k) selected as

$$q(i,k) = (1-\beta)\beta^{k-i} \quad (13)$$

where $0 < \beta < 1$. This weighting function penalizes the current estimation errors more than the past ones. It may be noted here that for large values of k

$$\sum_{i=0}^{k} q(i,k) = 1-\beta^{k+1} \approx 1 , \quad (14)$$

and therefore the results in Section 2 are applicable here. Then, the LS solutions work out to

$$\overset{0}{\hat{A}}(k) = \hat{R}_{xx}^{-1}(k)\ \hat{R}_{yx}(k) \tag{15}$$

and

$$\overset{0}{\hat{B}}(k) = \frac{1}{2}\ \hat{R}_{xx}^{-1}(k)\ T_{yx}(k)\ \hat{R}_{xx}^{-1}(k) \tag{16}$$

where

$$\hat{R}_{yx}(k) = (1-\beta) \sum_{i=0}^{k} \beta^{k-i} y(i)X(i). \tag{17}$$

$$\hat{R}_{xx}(k) = (1-\beta) \sum_{i=0}^{k} \beta^{k-i} X(i)X^T(i) \tag{18}$$

and

$$T_{yx}(k) = (1-\beta) \sum_{i=0}^{k} \beta^{k-i} y(i)X(i)X^T(i)\ . \tag{19}$$

To obtain the recursive relationships, we proceed as follows: Substituting (17) in (15) and simplifying, we get

$$\hat{R}_{xx}(k)\hat{A}(k) = (1-\beta) \sum_{i=0}^{k-1} \beta^{k-i} y(i)X(i) + (1-\beta)y(k)X(k) \tag{20}$$

and

$$\left[(1-\beta) \sum_{i=0}^{k-1} \beta^{k-i} X(i)X^T(i)\right] \hat{A}(k-1) = (1-\beta) \sum_{i=1}^{k-1} \beta^{k-i} y(i)X(i)\ , \tag{21}$$

where we have used $\hat{A}(k)$ and $\hat{B}(k)$ instead of $\overset{0}{\hat{A}}(k)$ and $\overset{0}{\hat{B}}(k)$. Substituting (21) in (20) and manipulating the resultant equation, we obtain the recursive relationship for $\hat{A}(k)$ as

$$\hat{A}(k) = \hat{A}(k-1) + (1-\beta)\ \hat{R}_{xx}^{-1}(k)X(k)\ \left[y(k) - X^T(k)\hat{A}(k-1)\right]\ . \tag{22}$$

The derivation of the recursive relationship for the quadratic weights is more involved, even though straightforward. We will give only the final result. For details, the reader is referred to [3]. $\hat{B}(k)$ is related to $\hat{B}(k-1)$ as

$$\hat{B}(k) = \frac{1}{\beta}\left[I-(1-\beta)\hat{R}_{xx}^{-1}(k)X(k)X^T(k)\right]\hat{B}(k-1)\left[I-(1-\beta)X(k)X^T(k)\hat{R}_{xx}^{-1}(k)\right]$$

$$+ \frac{1}{2}(1-\beta)y(k)\hat{R}_{xx}^{-1}(k)X(k)X^T(k)\hat{R}_{xx}^{-1}(k)\ , \tag{23}$$

where I denotes the NxN identity matrix. Also, $\hat{R}_{xx}^{-1}(k)$ is computed using the recursive relationship [3]

$$\hat{R}_{xx}^{-1}(k) = \frac{1}{\beta}\ R_{xx}^{-1}(k-1) - \frac{1}{d(k)}\left[\frac{1-\beta}{\beta^2}\ \hat{R}_{xx}^{-1}(k-1)X(k)X^T(k)\hat{R}_{xx}^{-1}(k-1)\right] \tag{24}$$

where

$$d(k) = 1 + \frac{1-\beta}{\beta} X^T(k)\hat{R}_{xx}^{-1}(k-1)X(k) \tag{25}$$

and d(k) is assumed to be nonzero. In deriving (24) we made use of the fact that

$$\hat{R}_{xx}(k) = \beta\hat{R}_{xx}(k-1) + (1-\beta)X(k)X^T(k) \tag{26}$$

and then used the matrix inversion lemma [7] which says that if A = B + CD,

$$A^{-1} = B^{-1} - \frac{1}{1 + DB^{-1}C} B^{-1}CDB^{-1}$$

with $1 + DB^{-1}C \neq 0$.

In the next section, we will demonstrate the usefulness of the proposed algorithm with a simulation example.

4. A SIMULATION EXAMPLE

To study the performance of the proposed algorithm, we consider the system identification problem for a second order Volterra filter whose coefficients A and B are given by

$$A = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.2 \end{bmatrix} \tag{27}$$

and

$$B = \begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{bmatrix} = \begin{bmatrix} 0.3 & -0.1 \\ -0.1 & 0.15 \end{bmatrix} . \tag{28}$$

Note that B is symmetric. The system identification problem is schematically depicted in Fig. 1, where the reference input x(k) is white, Gaussian with zero mean and unit variance and y(k) is obtained from x(k) using (2) with A and B given by (27) and (28), respectively. The algorithm was initialized with

$$\hat{A}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\hat{B}(0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ and}$$

$$\hat{R}_{xx}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} .$$

Thirty independent simulations were run using 2000 data samples each and the results presented are averaged over the thirty runs. Figure 2a-2e display plots of $\hat{a}_0(k)$, $\hat{a}_1(k)$, $\hat{b}_{00}(k)$, $\hat{b}_{01}(k) = \hat{b}_{10}(k)$ and $\hat{b}_{11}(k)$, respectively. In each case, we can see that the algorithm converges to the correct values. Also, the plot of the squared estimation error as a function of time in Fig. 2f shows that the extimation error decreases exponentially with time.

5. SUMMARY AND CONCLUSIONS

The adaptive second order Volterra filter with the SER algorithm was developed in this paper. The recursive algorithm tracks the LS solution that minimizes the given cost functional. The simulation results presented demonstrated the effectiveness of the algorithm in estimating the system parameters of a second order Volterra filter.

The Gaussian assumption for the reference input reduces the computational load considerably over the direct approach given by (12). The theoretical convergence properties of the adaptive filter are currently under investigation and will be the subject of a later paper.

REFERENCES

1. M. Schetzen, The Volterra and Wiener Theories of the Nonlinear System, Wiley, New York, 1980.

2. N. Ahmed, et al., "A Short-term Sequential Regression Algorithm," IEEE Trans. Acoust. Speech, Signal Proc., Vol. ASSP-27, No. 5, October 1979.

3. K.K. Yu, "Nonlinear Adaptive Digital Filter with Sequential Regression (SER) Algorithm," Master's Thesis, University of Iowa, December 1984.

4. T. Koh and E.J. Powers, "An Adaptive Nonlinear Digital Filter with Lattice Orthogonalization," Proc. ICASSP 83, pp. 37-40, Boston, 1983.

5. T. Koh, E.J. Powers and R.W. Miksad, "An Approach to Time Domain Modelling of Nonlinear Drift Oscillations in Random Seas," Proc. Int. Symp. Offshore Engineering, Rio de Janeiro, Brazil, 1983.

6. D. Graupt, Identification of Systems, Van Nostrand, 1972.

7. D.K. Faddeeva and V.N. Faddeeva, Computational Methods of Linear Algebra, W.H. Freeman and Co., San Francisco, 1963.

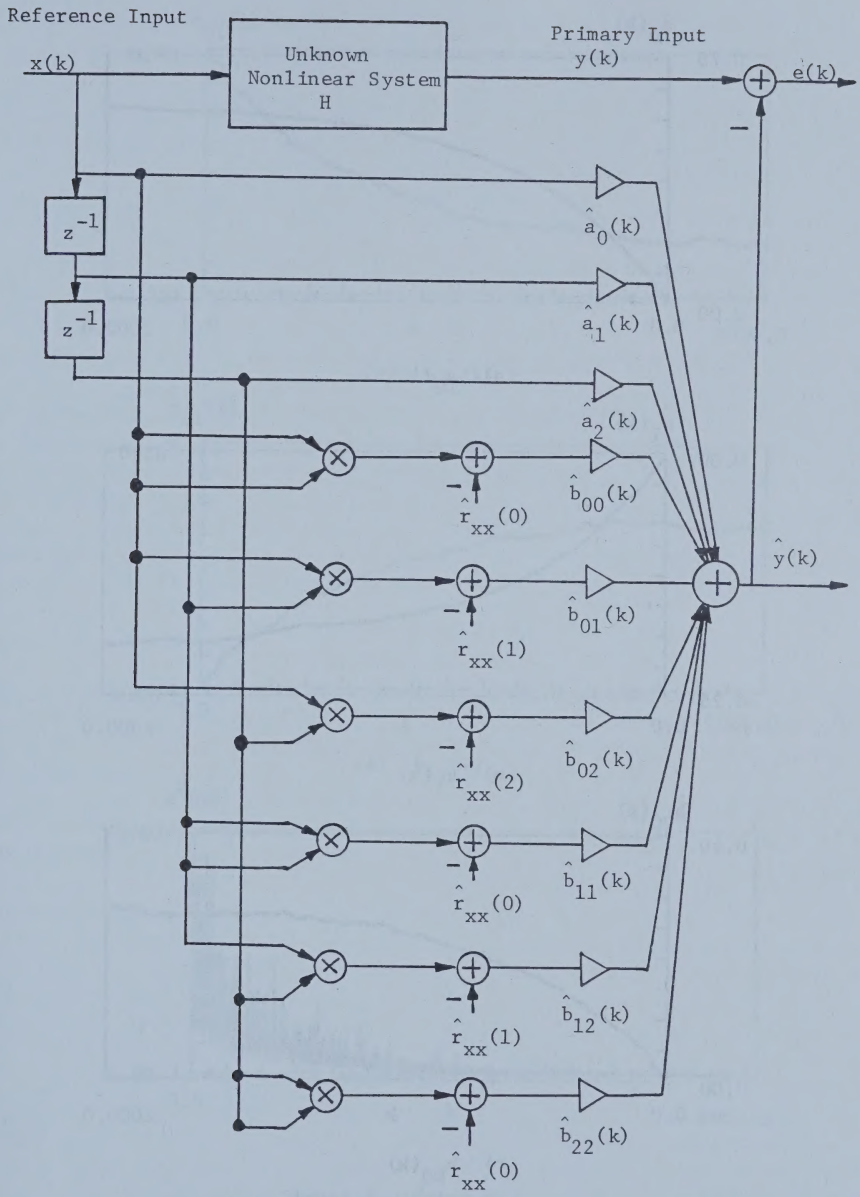


Figure 1. Tapped delay line model realization of a second order Volterra filter for the system identification problem.

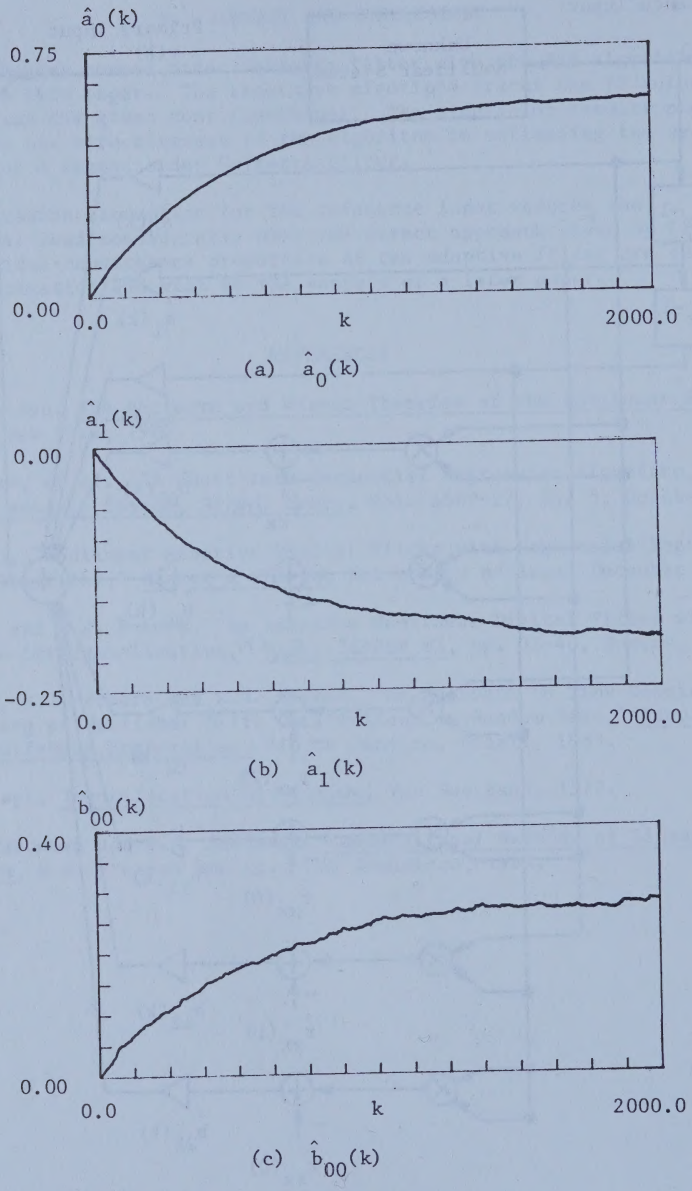
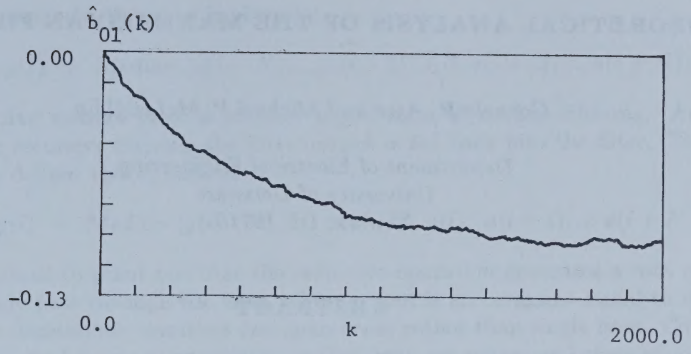
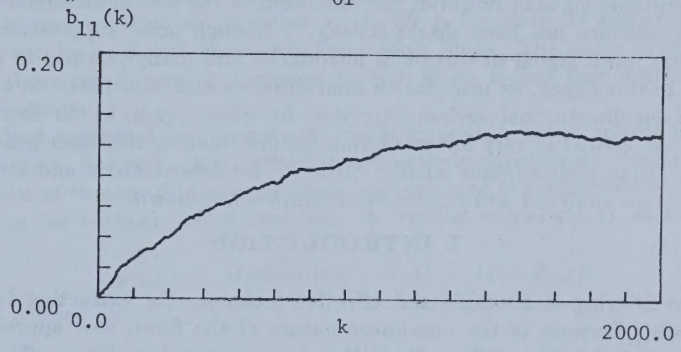


Figure 2. Ensemble averages of the system identification problem in the simulation example (β = 0.998).



(d) $\hat{b}_{01}(k)$



(e) $\hat{b}_{11}(k)$

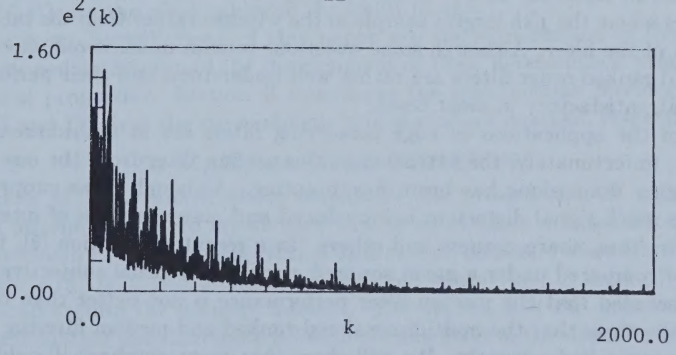


Figure 2. (cont.)

THEORETICAL ANALYSIS OF THE MAX/MEDIAN FILTER *

Gonzalo R. Arce and Michael P. McLoughlin

Department of Electrical Engineering
University of Delaware
Newark, DE 19716

ABSTRACT

Median filtering has been used successfully for extracting features from noisy one dimensional signals; however, the extension of the one dimensional case to higher dimensions has been disappointing. Although noise suppression is obtained, too much signal distortion is introduced and many features of interest are lost. In this paper, we introduce a multi-dimensional filter based on a combination of one dimensional median estimates. Invariant signals to the filter, called root signals, consist of very low resolution features making this filter much more attractive than conventional median filters. The deterministic and statistical properties are analyzed and illustrative examples are shown.

I. INTRODUCTION

Median filtering is a simple and effective technique for extracting features from noisy signals. Because of the non-linear nature of the filter, new approaches to the study and understanding of these filters have been undertaken. These efforts have been very successful, especially in the understanding of one dimensional median and ranked order filters where the $j'th$ largest sample in the window rather than the median is set as the output of the filter. Although some questions remain unanswered, one dimensional median and ranked order filters are rather well understood and their performance is, if not optimal, satisfactory in most cases.

Most of the applications of edge preserving filters are in multidimensional signal processing; unfortunately, the extension of the median filter from the one dimensional case to higher dimensions has been disappointing. Although noise suppression is obtained, too much signal distortion is introduced and many features of interest are lost, such as thin lines, sharp corners and others. In a recent publication [3], four different filters were compared under a mean squared error and a visual subjective criteria; the author concluded that the median filter performance is not better than of the others. These results show that the multidimensional ranked and median filtering problem has not been approached correctly. We will show that more emphasis should be directed towards structural information; hence, the filters should be designed so as to account for the multidimensional nature of the features embedded in the images. Before we define the Max/Median filter we review two dimensional implementations of the median filter as well as introduce common terminology.

To implement a non-recursive median filter, an odd number of sample values spanned by a window are sorted and the mid or median value is used as the filter output. If we let $\{a(.)\}$ and $\{y(.)\}$ be the input and output of the median filter with window size

* The authors gratefully acknowledge the support of the National Science Foundation under the grant No. ECS 830-7764

2N+1, respectively, then $y(.)$ is given by

$$y(i) = Median\ [a(i-N), ..., a(i-1), a(i), a(i+1), ..., a(i+N)].$$

The recursive median filter is another useful form of median filtering. As the name implies, in recursive filtering the filter output is fed back into the filter. The following expression defines this operation.

$$y(i) = Median\ [y(i-N), ...,\ y(i-2),\ a(i),\ a(i+1), ...\ a(i+N)].$$

It is important to point out that the recursive operation generates a root output with only a single pass through the data, where a root is an invariant signal to the filter.

In two dimensions, windows can span areas rather than single lines. Common window shapes used in image processing applications are square and cross windows. Again the filtering process consists in ranking the samples within the area of the window and choosing the median value as the output of the filter. Recursive as well as non-recursive operations can be used with these two dimensional filters. The major drawback associated with these two dimensional window median filters is that they destroy important features in images.

A method presented by Narenda [4] yields somewhat better results in preserving signal structure; however, this method is not entirely satisfactory. Separable median filters consist of two one dimensional filters, one oriented in the horizontal direction and the other in the vertical. More explicitly, the output value, $y(i,j)$, at position i,j is given by

$$y(i,j) = Median\ [z(i-N,j), ..., z(i+N,j)],$$

where z is defined as

$$z(p,q) = median\ value\ of\ [a(p,q-N), ..., a(p,q+N)],$$

where $\{a(i,j)\}$ are the pixel values of the input image.

The primary contributions of this paper are the introduction of the multidimensional max/median filter, and the development of basic deterministic properties as well of statistical properties. Section II introduces the max/median filtering process, and section III and IV gives the deterministic and statistical analysis, respectively.

II. MAX/RANKED FILTERING

The class of Max/ranked filters emphasize the preservation of multidimensional structural information based on the fact that in most cases the desired features are of less dimensionality than the observation space. For instance, consider the estimation of a sample a(i,j) in a two dimensional sampled space $\{a(.,.)\}$ on a 2N+1 by 2N+1 neighborhood around the point of interest. The approach of the max/ranked filtering is that of estimating one dimensional features embedded in the multidimensional space. In fact, we trace all possible lines through the center sample and make a ranked operation on the samples lying along each line, separately. Combining all one dimensional estimates, we make the multidimensional estimate. The advantage of this approach is the preservation of signal features that have lower dimensionality than the observation space, such as lines in a two dimensional space and planes in a three dimensional space. Hence in general, the max/ranked filter output in an N dimensional space is defined as

$$y_{i_1,i_2,...,i_N} = Max(Z_1, Z_2, ..., Z_K)\ ,$$

where the Z_i's are the one dimensional ranked feature estimates

$$Z_j = \varphi[\ all\ a's\ spanned\ by\ the\ j'th\ line\ ].$$

Here, K is finite and depends on the dimension of the observation space and the size of the one dimensional window, and φ is a ranked operation such a the median or an M-type estimator. The maximum operation is chosen with the criteria to preserve edges and discontinuities in the filtered signals. If other ranks are chosen rather than the maximum it can be easily shown that the filter will tend to blurr edges of the original signal. In many applications all the $Z_j's$ estimates may not be needed. In two dimensions, for a filter of size 2N+1 the number of $Z_j's$ can be shown to be bounded by 4N(N-1). For instance, a subclass of these filters is one where we only take into account samples in lines separated by 45 degrees and the ranked operation is a median filter. Hence, denoting the input and output of the 2N+1 Max/Median filter as $a_{i,j}$ and $y_{i,j}$, respectively. Then the output of the max/median filter at the i,j position is defined as

$$y(i,j) = Max[\ Z_1, Z_2, Z_3, Z_4\ ]$$

where

$$\begin{aligned} Z_1 &= Median\ (a_{i,j-N}, ..., a_{i,j}, ..., a_{i,j+N}), \\ Z_2 &= Median\ (a_{i-N,j}, ..., a_{i,j}, ..., a_{i+N,j}), \\ Z_3 &= Median\ (a_{i+N,j-N}, ..., a_{i,j}, ..., a_{i-N,j+N}), \\ Z_4 &= Median\ (a_{i-N,j-N}, ..., a_{i,j}, ..., a_{i+N,j+N}). \end{aligned} \tag{2.1}$$

Hence this Max/Median filter of window size 2N+1 spans 8N+1 sample points. The window is moved from left to right and from top to bottom. Moreover, when filtering the samples at the edges, we append points to the edges equal to the value of the edge points. The next figures will illustrate the feature preserving properties of this filter and will also compare them with other common two dimensional median filtering techniques. Figures 1a and 1b show an original and noisy images, respectively. The noise is additive white normally distributed. The images were designed to best illustrate the feature preservation property of the max/median filter for any window size and image resolution. Figures 2a and 2b show the separable non-recursive median filter of size 3 and the max/median filter of the size size, respectively. If we increase the window size of the filter, we can expect rapid deterioration of the results obtained with the separable filter as shown in Fig.3a. However, the max/median filtered image is not perturbed and it even seems to give a better estimate.

III. DETERMINISTIC PROPERTIES

We will now look at some of the deterministic properties of the max/median filter. First we state threshold decomposition [5] for multilevel signals and then some definitions of important signal structures for binary signals. Then by using the techniques of threshold decomposition these definitions will be extended to the case of multilevel signals. Finally some properties will be shown for both the standard and recursive max/median filters.

We define a multilevel signal $a(m,n)$ as being discrete and consisting of k levels. The thresholded signal is defined by

$$t_o^i(m,n) = \begin{cases} 1, & \text{if } a(m,n) \geq i \\ 0, & \text{if } a(m,n) < i \end{cases} \tag{3.1}$$

where $1 \leq i \leq k-1$ is the level being thresholded, and (m,n) is the point under consideration. The following two theorems will allow us to relate k binary sequences to a k level signal. The proofs of these theorems are not included here because of space limitaion. First for the standard max/median filter we have:

Theorem 3.1: If we threshold the signal as described in Equation 3.1 and apply the standard max/median filter to the thresholded binary sequences, we can relate the

multilevel output

$$y_s(m,n) = \max\{z_1(m,n), z_2(m,n), z_3(m,n), z_4(m,n)\}$$

to the filtered binary sequences by

$$y_s(m,n) = \max\{\max(0, i : x^i_{s,1}(m,n) = 1), \ldots, \max(0, i : x^i_{s,4}(m,n) = 1)\}.$$

And similarly for the recursive max/median filter:

Theorem 3.2: If a k level signal is thresholded as described in Equation 3.1, and the binary sequences filtered with the recursive max/median filter, then the output

$$y_r(m,n) = \max\{z_1(m,n), z_2(m,n), z_3(m,n), z_4(m,n)\}$$

can be constructed from the binary sequences by

$$y_r(m,n) = \max\{\max(0, i : x^i_{r,1}(m,n) = 1), \ldots, \max(0, i : x^i_{r,4}(m,n) = 1)\}.$$

Where the $x^i_{s,j}$'s and $x^i_{r,j}$'s are the standard and recursively median filtered binary sequences with j indicating the window orientation and i the threshold level. ■

We now define some important signal structures for binary signals.

A **Binary Constant Neighborhood** (BCN) is a region of ones that are invariant to the max/median filter, consisting of each point being attached to at least N points of equal value along the same direction. (See Fig. 4a)

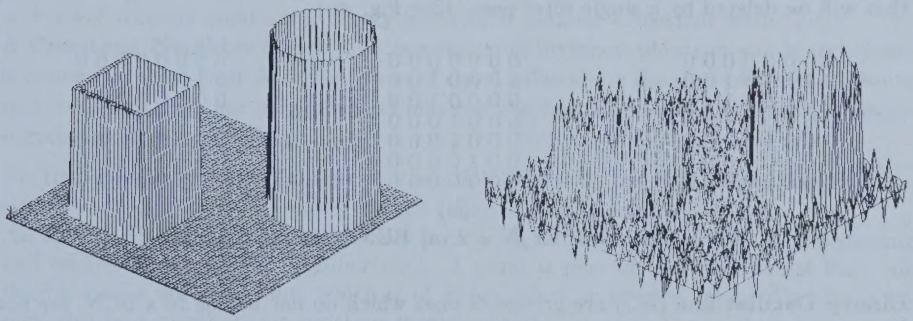


Fig.1 a) Input image. b) Noisy Image.

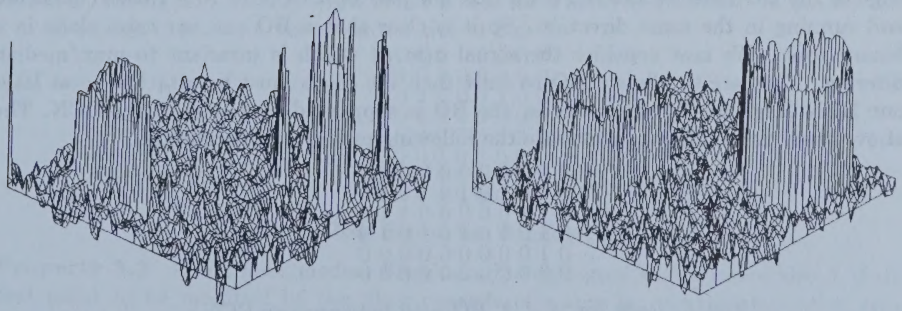


Fig.2 Filtered Images: a) Separable Median. b) Max/Median. (N=1)

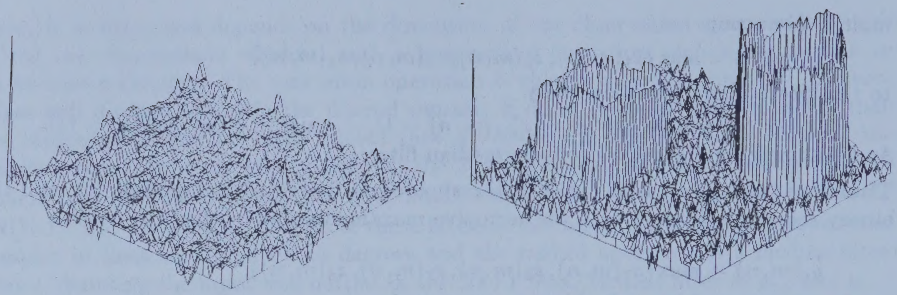


Fig.3 Filtered Images: a) Separable Median. b) Max/Median. (N=2)

Binary constant neighborhoods must be separated by at least $n+1$ zeros in any direction. In other words, any zero bordering the exterior edge of either of the BCN's must have at least n zeros attached to it along all directions to guarantee that it will not merge with any other regions. However, it is not difficult to devise cases were a smaller distance separates the two regions. Similarly if there is a region of zeros enclosed by the constant neighborhood then each zero must be connected to at least n other zeros along any direction to guarantee that the inner region will not be filled.

A **Binary Impulse** is a grouping of ones surrounded by zeros (as described for BCN's) that will be deleted by a single filter pass. (See Fig. 4b)

0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 1 1 1 1 1 0 0	0 0 0 0 1 0 0	0 1 0 1 0 1 0 0
0 0 1 1 1 1 1 0	0 0 0 0 1 0 0	0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0	0 0 0 1 0 0 0	0 1 0 1 0 1 0 0
0 0 0 1 0 0 0 0	0 0 0 1 0 0 0	0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0	0 0 1 0 0 0 0	0 1 0 1 0 1 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
(a)	(b)	(c)

Fig. 4. Signal structures for $N = 2$: a) BCN b) binary impulse c)BO.

Binary Oscillations (BO) are groups of ones which do not belong to a BCN, are not impulses, and do not contain any regions with more than N ones running in the same direction. (See Fig. 4c)

A **Binary Root** is a binary signal which is invariant to max/median filtering.

Before looking at the nature of a binary root, let us first consider a BO. By definition a BO contains no BCN's. If the BO is surrounded by a region of zeros then a one on the edge of the BO must be deleted since it is not part of a BCN or $N+1$ ones connected and running in the same direction. So it is clear that a BO can not exist alone in a binary root. We now consider the signal $a(m,n)$ which is invariant to max/median filtering. If we are to allow a BO to exist then the edges must be attached to at least one BCN (Fig. 5). In other words, the BO is supported by points in the BCN. The above observation allows us to state the following property of oscillations.

0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 1 0 1 0 1 0 0
0 1 1 0 0 0 0 0 1 1 0 0
0 1 0 1 0 1 0 1 0 1 0 0
0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0

Fig. 5. Root for $N = 2$, BO held between two BCN's

Property 3.1 In a root signal it is necessary for any binary oscillation to be attached to at least one binary constant neighborhood. ∎

Again consider the invariant signal $a(m,n)$. We next examine the point (i,j). If this point is a zero then there can be no more than n ones along any filter direction. If there were more than this the output of the max/median filter would increase the signal at this point. So we can see that for a zero to be invariant to max/median filtering it is a necessary and sufficient condition that the median value of each of the four windows be zero.

Next let us consider a point (p,q) which is a one. If the output of the max/median filter at this point is to be a one then there must be at least $n+1$ ones in at least one of the window directions. Thus the point (p,q) can be part of either a BCN or a BO. If the point is part of a BO, the BO must be attached to at least one BCN since by Property 3.1 a BO can not exist by itself. Thus we have proved the following theorem.

Theorem 3.3 For a binary signal to be a binary root to the max/median filter it is sufficient that it be composed of binary constant neighborhoods and binary oscillations.

Now we will use the results of Theorems 3.1 and 3.2 to analyze the k level filtering problem. If the k level signal is a root to the max/median filter it is easy to see that the thresholded sequences obtained from (3.1) are also invariant. Thus we can construct a k level root by combining binary roots in a manner consistent with (3.1).

A **Constant Neighborhood** (CN) is a region of invariant points in which each point is connected to at least N other points of equal value along the same direction. Also, a region is not a constant neighborhood unless it can be constructed by combining binary signals consisting of BCN's.

An **Impulse** is a region of points that will be deleted by the max/median filter and is constructed by combining binary signals consisting of binary impulses.

An **Edge** is a series of monotonically increasing or decreasing points both beginning and ending in a constant neighborhood. A point is part of edge if it has at least one neighbor of a different value which is of either lesser or greater value. An edge point may also be a member of a constant neighborhood.

A **Oscillation** is a point which is not part of a constant neighborhood, impulse or edge.

A **Root** is a k level signal which is invariant to the max/median filter.

It has been shown [1] that in the case of the one dimensional median filter, a root signal for a window size of $2N+1$ points is also a root for any smaller window size. However, the signal shown in Fig.6 is not a root for a max/median filter of window size 3 even though it is a root for a window size of 5 or 7.

0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0
0 0 2 0 0 0 1 0 0
0 0 0 3 0 3 0 0 0
0 0 0 0 3 3 3 3 0
0 0 0 3 0 3 0 0 0
0 0 3 0 0 0 3 0 0
0 0 0 0 0 0 0 0 0

Figure 6. Root Signal for Window Size 5 or 7

Property 3.2: When max/median filtering a binary signal with window size 3, if the first point to be modified by the filter consists of a zero to one transformation then the point becomes invariant. Furthermore, until the first one to zero transformation is encountered all modified points will be invariant.

Proof: In order for a point (i,j) to make a zero to one transformation there must be

two ones running in at least one direction. For a window size of three this means that both of these points must be adjacent to the point (i, j). Because no points up to this point have made a one to zero transformation, the point (i, j) must become invariant since it will have at least one neighbor that equals one on the next pass. As long as no ones are deleted by the filter it is not hard to see that the above will hold for the following points. ■

Property 3.3: After repeated filtering, a binary image image will be bordered by an invariant region.

Proof: Because we append points to the edge of an image, a one on an edge is invariant. Therefore, it is impossible for edge points to be part of any type of vibration region and the edges must eventually become invariant. ■

Theorem 3.4: If a k level signal is repeatedly filtered with a max/median filter of window size 3, the filtered signal will converge to a root.

Proof: First we consider a binary signal. By property 3.3 we know that edge points must become invariant after repeated filtering. If we consider the first non-edge point in our two dimensional signal then by property 3.2 this point can not oscillate indefinitely. Thus it will either converge to a one or a zero. By applying an inductive argument it can be seen that all points must eventually become invariant. Therefore a binary signal will become invariant after repeated filter passes. If we threshold the original signal into k binary sequences as described by (3.1) and filter each repeatedly by the above argument each must become invariant. Therefore the k level signal must also become invariant. ■

While we have not shown that a two dimensional signal will converge to a root for any window size we speculate that Theorem 3.4 will hold in general. After running many simulations we have not yet discovered a signal that would not converge. If we first replace the point (i, j) with the output of the max/median filter before shifting the window to the next point we have a recursive max/median filter. Here we will investigate some of the properties of the recursive filter and then prove that an arbitary signal must converge to a root signal for window size of 3 or 5. It can be shown that any signal that is invariant to standard max/median filtering is also invariant to recursive max/median filtering. Moreover, a signal that is a root for a window size (2N+1) is not in general a root for a filter of smaller window size.

Property 3.4: When max/median filtering a signal with a window size of 3 or 5, once a point (i, j) is increased by the max/median filter it may never be decreased again.

First let us consider the thresholded binary signal. For a point located at (p, q) to change from a zero to a one there must be at least $n + 1$ ones running in at least one direction. For a window size of 3 or 5 this means that the output point must be a part of $N + 1$ ones running in the same direction. Thus we see that once a point in the binary plane undergoes a zero to one transformation it becomes invariant.

For the k level signal to increase from i to $i + l$ at point (p, q), the thresholded signals from $i + 1$ to $i + l$ must undergo a zero to one transformation. Since by the above argument these points are invariant, the point (p, q) can not decrease from this level. ■

Theorem 3.5 An arbitrary signal will converge to a root signal after repeated passes of the recursive max/median filter of window size 3 or 5.

Proof: By property 3.3 we see that it is impossible for any point (p, q) to temporally oscillate. Hence the signal must eventually become invariant to further filtering. ■

As with the standard max/median filter we again speculate that Theorem 3.5 will hold for any window size $(2N + 1)$.

IV. OUTPUT DISTRIBUTIONS OF THE MAX/MEDIAN FILTER

In this section we will determine the output distribution function for the output of the max/median filter with parameter N (window size 2N+1), when the input consists of independent samples from a random variable X with distribution function $F_X(x) = P(X \leq x)$. The output of the max/median filter is $y_{i,j} = max(Z_1, Z_2, Z_3, Z_4)$, where the variables Z's are defined in (2.1). Define the M sets as

$$M_1 = (\textit{ samples generating } Z_1 \textit{ and excluding } x_{i,j})$$

$$M_2 = (\textit{ samples generating } Z_2 \textit{ and excluding } x_{i,j})$$

$$M_3 = (\textit{ samples generating } Z_3 \textit{ and excluding } x_{i,j})$$

$$M_4 = (\textit{ samples generating } Z_4 \textit{ and excluding } x_{i,j}).$$

Then the first order output distribution function is $F_Y(y) = F_{Z_1,Z_2,Z_3,Z_4}(y,y,y,y)$,or

$$P\Big(y_{i,j} \leq y\Big) = P(Z_1 \leq y, Z_2 \leq y, Z_3 \leq y, Z_4 \leq y).$$

Define the event

$$B(K, \Theta, y) = (\textit{at least } K \textit{ samples in the set } \Theta \textit{ are } \leq y),$$

then the output distribution function can be written as

$$P(y_{i,j} \leq y) = P\Big(B(N+1,\ M_1 \cup x_{i,j}, y),\ B(N+1,\ M_2 \cup x_{i,j}, y),$$

$$B(N+1,\ M_3 \cup x_{i,j}, y),\ B(N+1,\ M_4 \cup x_{i,j}, y) \Big).$$

Using the law of total probability we can write this function as

$$P(y_{i,j} \leq y) = \Big[P\Big(B(N+1,\ M_1 \cup x_{i,j}\ , y),\ \ B(N+1,\ M_2 \cup x_{i,j}\ , y),$$

$$B(N+1,\ M_3 \cup x_{i,j}, y),\ B(N+1,\ M_4 \cup x_{i,j}, y)\ /\ x_{i,j} \leq y\Big)\ P\Big(x_{i,j} \leq y\Big)\Big]$$

$$+\ \Big[P\Big(B(N+1,\ M_1 \cup x_{i,j}\ , y),\ B(N+1,\ M_2 \cup x_{i,j}\ ,\ B(N+1,\ M_3 \cup x_{i,j}, y),$$

$$B(N+1,\ M_4 \cup x_{i,j}, y)\ /\ x_{i,j} > y\Big)\ P\Big(x_{i,j} > y\Big)\Big]$$

or

$$P(y_{i,j} \leq y) = P\Big(B(N, M_1, y), B(N, M_2, y), B(N, M_3, y), B(N, M_4, y)\Big) F_X(y)$$

$$+\ P\Big(B(N+1, M_1, y),\ B(N+1, M_2, y),\ B(N+1, M_3, y),\ B(N+1, M_4, y)\ \Big)[1 - F_X(y)].$$

Since samples from different sets $M's$ are mutually exclusive, then the first order distribution for i.i.d. inputs is

$$F_Y(y) = \left[\sum_{j=N}^{2N} \binom{2N}{j} F_X^j(y)\,[1 - F_X(y)]^{2N-j}\right]^4 F_X(y)$$

$$+\quad \left[\sum_{j=N+1}^{2N} \binom{2N}{j} F_X^j(y)\,[1 - F_X(y)]^{2N-j}\right]^4 [1 - F_X(y)]. \qquad (4.1)$$

As an example consider the filtering of i.i.d. uniform and exponentially distributed noise. Figure 7 shows the density of the filtered variables using a cross median filter of window 3 and a max/median filter of the same size. Clearly the max/median filter produces asymmetry in the density on the density of the filtered variates. Because of the maximum operator applied on the Z variables, we expect the output to be a biased estimator of the signal. This is shown in the following property.

Property 4.1 The output of the max/median filter is a median biased estimator.

This can be seen as follows. Recall that the i.i.d. input samples have a distribution function $F_X(x) = P(X \leq x)$. Let the median of the input samples be $\tilde{x}$; hence, $F_X(\tilde{x}) = 0.5$. Then the output distribution at the median $\tilde{x}$ is

$$F_Y(\tilde{x}) = \left[\left(\sum_{j=N}^{2N} \binom{2N}{j} \right)^4 + \left(\sum_{j=N+1}^{2N} \binom{2N}{j} \right)^4 \right] \left(\frac{1}{2}\right)^{8N+1}.$$

Since

$$\sum_{j=0}^{2N} \binom{2N}{j} = 2^{2N}, \text{ and } \binom{2N}{j} = \binom{2N}{2N-j},$$

then

$$F_Y(\tilde{x}) = \left[\left(\frac{2^{2N}}{2} + \frac{1}{2}\binom{2N}{N} \right)^4 + \left(\frac{2^{2N}}{2} - \frac{1}{2}\binom{2N}{N} \right)^4 \right] \left(\frac{1}{2}\right)^{8N+1}. \tag{4.2}$$

But,

$$\binom{2N}{N} = \left[\frac{2^N.(2N-1).(2N-3)...3.1}{N!} \right] < 2^{2N}.$$

Replacing the above inequality in (4.2) we get

$$F_Y(\tilde{x}) < \frac{1}{2}.$$

Hence, the median of the output y is greater than the median of the input data.

It is also of interest to have an idea of the effect of filter on the mean of the original data. In most cases, finding the mean of the output of the filter by use of equation (4.1) can be computationally very tedious and in many cases it can only be tabulated since closed form expressions are not obtained. However, bounds on the mean of the output can be obtained.

Property 4.2 Let μ_x, μ_y, μ_z be the expected values of the input, output, and Z variables, respectively. Define the standard deviation of the Z variables be σ_z, then

$$\mu_z \leq \mu_y \leq \mu_z + \sigma_z\sqrt{3}.$$

Moreover, if the input samples have a symmetric distribution, then

$$\mu_x \leq \mu_y \leq \mu_x + \sigma_z\sqrt{3}.$$

This can be seen as follows. It is shown in [5] that if Z_i (i=1,2,...k) are possible dependent variate, each with mean μ_z and variance σ_z^2. Then for any constants a_i

$$|E\left(\sum a_i \left(Z_{(i)} - \mu_z\right)\right)| \leq \left[k\, \sigma_z^2 \sum (a_i - a)^2\right]^{\frac{1}{2}}, \tag{4.3}$$

where $Z_{(i)}$ are the ordered variates. Letting $\mu_{Z_{(i)}}$ be the expected value of the ith ordered variate $Z_{(i)}$, and noticing that

$$\sum_{i=1}^{r} \mu_{Z_{(i)}} \leq r\, \mu_{Z_{(r)}}, \quad \text{and} \quad \sum_{i=r}^{k} \mu_{Z_{(i)}} \geq (k - r + 1)\, \mu_{Z_{(r)}},$$

then combining (4.3) with

$$\frac{1}{r}\sum_{i=1}^{r}\mu_{Z_{(i)}} \leq \mu_{Z_{(r)}} \leq \frac{1}{k-r+1}\sum_{i=r}^{k}\mu_{Z_{(i)}} ,$$

we get,

$$\mu - \sigma\left(\frac{k-r}{r}\right)^{\frac{1}{2}} \leq \mu_{Z_{(r)}} \leq \mu + \sigma\left(\frac{r-1}{k-r+1}\right)^{\frac{1}{2}}.$$

In the max/median filter case, $r = k = 4$; hence, property 4.2 follows.

As an example, if we filter i.i.d. samples distributed with an exponential distribution, ie. $f(x) = e^{-x}$ $, x \geq 0$ then the mean of the output of the filter with parameter N is bounded as

$$\sum_{k=1}^{N+1}\left(\frac{1}{2N+2-k}\right) \leq \mu_y \leq \sum_{k=1}^{N+1}\left(\frac{1}{2N+2-k}\right) + \sqrt{3\sum_{k=1}^{N+1}\left(\frac{1}{2N+2-k}\right)^2}.$$

For N=1 we have $0.9 \leq \mu_y \leq 1.4$ and for N=2 the bound is $0.8 \leq \mu_y \leq 1.14$.

V. CONCLUSIONS

The max/median filter is shown to be an effective filter for removing noise as well as preserving two dimensional image features. It was shown that for small window sizes, convergence of signals to roots is guaranteed. The max/median root structure is defined and it is shown that they have more flexibility than roots of two dimensional filters. From a statistical point of view, the first order distribution is obtained and it is shown that the max/median filter is a median biased estimator. Finally, bounds on the expected bias of the mean are derived.

VI. REFERENCES

[1] N.C. Gallagher and G.L. Wise . " A Theoretical Analysis of The Properties of Median Filters," *IEEE Transactions on ASSP*, Vol. ASSP-29, no. 6, December 1981.

[2] P.Fitch and N.C.Gallagher, Median Filtering by Threshold Decomposition. *Proc.of the 1984 Princeton Conference*, March 1984.

[3] S.Dravida, J.W.Woods, and W.C. Shen, "A Comparison of Image Filtering Algorithms," *Proc. of the 1984 IEEE ICASSP* Vol.2 ,March 1984.

[4] P.M. Narenda, "Noise Reduction by Median Filtering," *Proc. of the IEEE conf. on Pattern Recognition*, 1978.

[5] H.A. David,*Order Statistics*, sec. ed. Wiley, 1981.

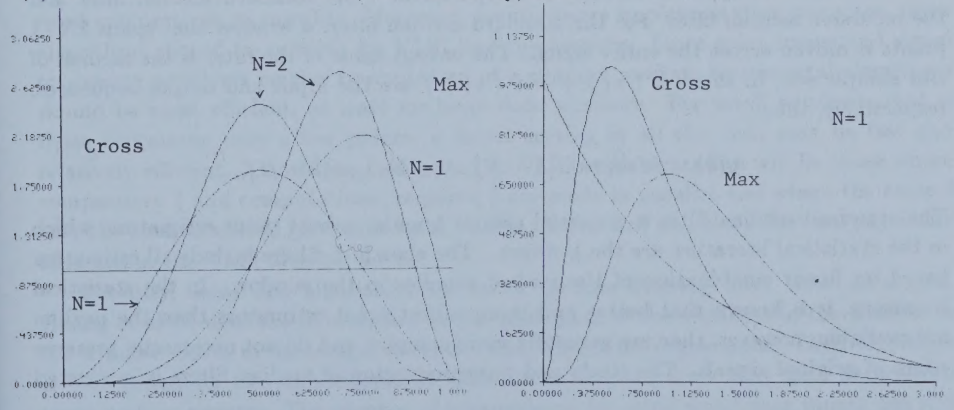


Fig.7 a) Density functions for Median and Max/Median Filtered Signals when input noise is: a) Uniform (0,1). b) Exponential

A MEDIAN FILTER ARCHITECTURE SUITABLE FOR VLSI IMPLEMENTATION*

G.R. Arce and P.J. Warter

Department of Electrical Engineering
University of Delaware
Newark, DE 19716

ABSTRACT

Median filtering is a nonlinear digital technique used in many applications for extracting features from noisy images. A scheme for implementing ranked recursive and non-recursive filters using a stored-ordered list is described and a first version in VLSI NMOS is analyzed for size and timing considerations. A comparison with an alternate systolic architecture is made. This algorithms scales very well for large window sizes. Real time video processing should be possible.

I. INTRODUCTION

Median filtering is a nonlinear digital technique used in many applications for extracting features from noisy signals. One of the important characteristics of the filter is that it behaves as a low pass filter, but at the same time preserves important signal structures such as edges; this property has proven very useful in image processing where edge information is very important. In general, median filters perform better than linear filters whenever impulsive or heavy tailed distributed noise is present [1]. Edge preserving digital filters are used in many applications such as computerized tomography, remote sensing, robot vision and medical imaging; hence, fast hardware implementations of these filters should prove very useful.

There are two types of median filter operations , the standard median filter and the recursive median filter. For the standard median filter, a window that spans 2N+1 points is moved across the entire signal. The output value of the filter is the median of this sample set. In short, if (x(.)) and (y(.)) are the input and output sequences, respectively, then

$$y(k) = Median[x(k-N), ..., x(k), ..., x(k+N)].$$

The standard median filter is a special case of popular robust point estimators, which in the statistical literature are the L-filters . The class of L-filters include all estimators based on linear combination of the ranked samples in the window. In the statistical literature, it is known that better and more robust point estimators than the median are available; however, they are generally more complex and do not necessarily preserve edges of original signals. The study and implementation of median filters is motivated

* **The authors gratefully acknowledge the support of the National Science Foundation under the grant No. ECS 830-7764**

by the combination of their simplicity and their effectiveness.

Recursive median filtering is a modification of the standard process which results in the output being the median of the last N outputs and the last N+1 inputs. In short notation, if (y(.)) and (x(.)) are the output and input sequences respectively, then

$$y(k) = Median[y(k-N), ..., y(k-1), x(k), ..., x(k+N)].$$

Hence, the recursive filter is an operation with memory and it produces a stable output in one pass that is one that does not change with further passes of the filter over the output, and often provides better image enhancement than a single pass of the standard median filter. The effect of the window size (ie. 2N+1) for median filters is simple, as the window is made larger the filter smooths the noisy signal better; however, more distortion is introduced in the original signal structure.

Empirically it is known that for a single filter pass, the recursive filter extracts signal features better than the non-recursive median filter. This fact has also been shown statistically [2], and it should be of consideration in the design of median filters for real time applications. Note that for the same input signal, the signals obtained with multiple passes of the standard median filter and of a single pass of the recursive median filter are not necessarily equal; however, in most cases they are very similar. Moreover, invariant signals to recursive filters are also invariant to standard median filter and viceversa. The relative merits of recursive and non-recursive filters have been discussed in [2,3].

In some cases, letting the output be the the kth largest point spanned by the window rather than the median will yield better results. These set of filters, which include the median filter, are denoted as rank order filters. Such rank order filters are useful in restoring signals with multiplicative noise and might also be used as peak or valley follower [3]. The implementation presented here can be programmed so as to yield any rank value as the output of the filter. In fact, with some added hardware any L-type filter can be implemented.

Because of the large amount of data processed in image processing applications and since conventional sorting methods, such as Quicksort [4], are time-consuming, there is a strong need for algorithms that search for sample medians and have as few word comparisons as possible. Moreover, for hardware implementation purposes, these algorithms should be suitable for VLSI implementations. Since most image and signal processing problems involve computation of a running median, incremental algorithms should be most efficient, at least for large data windows. For small windows, that is those containing only a few points, a direct sorting of all the data may be fast and relatively efficient. Algorithms best suited to VLSI implementation will be those where comparisons (and computations, required) are made in parallel and where the control over the computational process is local, that is, dependent on values that are physically adjacent in the VLSI filtering device.

Recently, some fast algorithms for faster median filtering have been developed. Huang et al.[5] used the information in the histogram of the elements in the window as the window moves in the image, to reduce the number of comparisons. Their algorithm is best suited for two dimensional median filters with windows that span determined areas such as squares. The number of comparisons for some worst-case inputs can still be too large and the method might be somewhat slow. Moreover, the memory requirement is large and since this method is inherently serial it is not well suited for VLSI

implementation. Ataman et al. [6] developed an elegant algorithm which determines the kth bit of the median by inspecting the k most significant bits of the samples. The hardware complexity and speed of this method depends heavily on the number of bits of each word and does not depend on the size of the window. A rather large memory is required and extensive computation is done with this method. Oflazer [7] implemented a systolic algorithm on VLSI based on compare-swap regular structures with data flow at the bit level. Although this implementation is for a fixed one dimensional window of size 5, in general the hardware complexity of this architecture depends heavily on the window size. We will describe this last implementation in more detail in later sections.

The median filter described in this paper is one which keeps an ordered list of the last 2N+1 inputs (or inputs and outputs in the case of a recursive filter). An input sequence number (mod 2N+1) is associated with each element in the list so that the oldest can be discarded when a new sample is added to the 2N+1 element list. Each sample value register in this design has a comparator so that a new sample can be compared with all of the current values at the same time. The opening of the list to allow insertion of the new value at the correct place and the discarding of the oldest value are accomplished by local control units associated with each register. This local control acts largely in response to the value comparisons made in the associated register and its neighbors. This design is relatively fast and adaptable. The window size and the rank of the element outputted can be programmed. The same filter can be programmed to be recursive or non-recursive.

II. MEDIAN FILTER ARCHITECTURES

A median filter must be able to store the 2N+1 samples in its window. They can be stored in order of arrival, or in value rank order. If they are stored in order of arrival, a comparison tree is required to order the samples as shown in Fig. 1a and as in the scheme proposed by Oflazer[7]. If they are stored in value order, it is necessary to tag each value with its order of arrival and provide circuits for inserting a new sample into its place in the list and removing the oldest sample. Such a system is shown in Fig. 1b. We have chosen the later architecture for the reasons outlined in the introduction.

Let us consider the problem of maintaining an ordered list. We must: 1) find the proper place for the new entry, 2) discard the oldest entry and 3) move the items in the list between the place for the new entry and the discarded entry. There are three cases: 1) the new item replaces directly the item discarded, 2) the place for the new item is above the place of the discarded item in the list, and 3) the place of the new item is below the place of the discarded item. These are shown in Fig. 2. Note that the sequence number for the new sample is the same as that of the sample being discarded (mod 2N+1). Note also that in making comparisons between the new sample and those in the value registers, we arbitrarily associate the equal condition with less than, and we arbitrarily assume that the largest value is at the "top" of the value ordered register chain and the smallest at the "bottom".

Consider the case when the window is centered at the ith location, that is the samples within the window are $x_{i-N}, ..., x_i, ..., x_{i+N}$. Each sample x_j ,for $j = i-N, ..., i+N$ is associated with an input sequence number $(j)_{mod2N+1}$; then, from the samples within the window we define the 2N+1 length register vector $\mathbf{X}_i$ and the 2N+1 length input

sequence vector $\mathbf{R}_i$ as

$$\mathbf{X}_i = \begin{pmatrix} x_{(2N+1)} \\ x_{(2N)} \\ \vdots \\ x_{(1)} \end{pmatrix}$$

and

$$\mathbf{R}_i = \begin{pmatrix} \alpha_{2N+1} \\ \alpha_{2N} \\ \vdots \\ \alpha_1 \end{pmatrix}$$

where $x_{(j)}$ is the jth largest sample among the 2N+1 samples in the window, and where α_j is the number $(l)_{mod 2N+1}$ associated with the jth largest value in the set such that

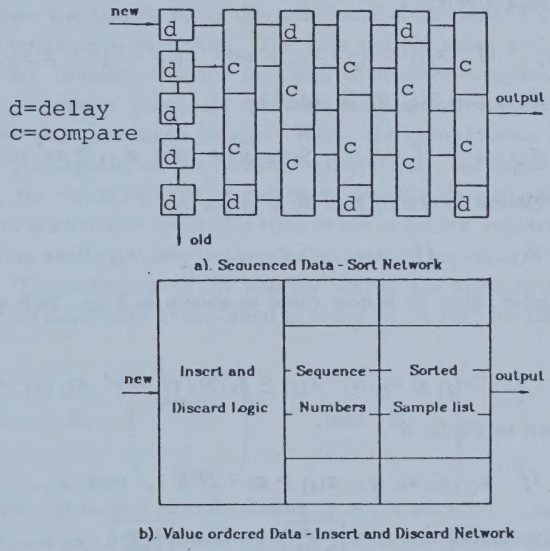


Fig.1 Two Approaches- Median Filter Architecture

$x_l = x_{(j)}$. As an illustration consider a window of size 5 centered at i=5 and with the samples in the window being $(x_3, x_4, x_5, x_6, x_7) = (2, 2, 5, 4, 3)$. The ranked vector and input sequence vector in this case are

$$\mathbf{X}_5 = \begin{pmatrix} x_{(5)} \\ x_{(4)} \\ x_{(3)} \\ x_{(2)} \\ x_{(1)} \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{pmatrix}$$

and

$$\mathbf{R}_5 = \begin{pmatrix} \alpha_5 \\ \alpha_4 \\ \alpha_3 \\ \alpha_2 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} (5)_{mod\ 5} = 5 \\ (6)_{mod\ 5} = 1 \\ (7)_{mod\ 5} = 2 \\ (3)_{mod\ 5} = 3 \\ (4)_{mod\ 5} = 4 \end{pmatrix}$$

Clearly, the output of the jth ranked filter is $y_i = x_{(j)}$ obtained from the $\mathbf{X}_i$ vector and from the input sequence vector we are able to tell which sample (register) is discarded as we move the window.

From an inspection of Fig. 2. it is obvious that on updating there are four things that can happen to each of the sample-sequence register pairs: 1) they can retain their previous values, 2) they can take on the values previously in the register pair above, 3) they can take on the values previously in the register pair below, or 4) the value register can be loaded with the new sample and the sequence register with its sequence number (mod 2N+1). More specifically if we let the jth element of $\mathbf{X}_i$ and of $\mathbf{X}_{i+1}$ be $x_{(j)}$ and $x'_{(j)}$, respectively, then as we move the window from position i to i+1 the registers will be modified according to the following equations where $x_{(j)}, x'_{(j)}, x_{i-N}$,and x_{i+N+1} are the jth value of $\mathbf{X}_i$, the jth value of $\mathbf{X}_{i+1}$, the sample being discarded , and the new sample introduced, respectively:

$$x'_{(j)} = x_{(j)} \quad if \quad x_{(j)} > (<)\ x_{i-N} \ \ and \ \ x_{(j)} > (<)\ x_{i+N+1}.$$

The register shifting shown in Fig. 2b is given by

$$x'_{(j)} = x_{(j+1)} \quad if \quad x_{(j+1)} \leq x_{i+N+1} \ \ and \ \ x_{(j)} \geq x_{i-N}.$$

and the shifting of registers shown in Fig. 2c is given by

$$x'_{(j)} = x_{(j-1)} \quad if \quad x_{(j-1)} > x_{i+N+1} \ \ and \ \ x_{(j)} \leq x_{i-N}.$$

The case when a register takes on a new value as shown in Figs. 2a,b is given by the equations

$$x'_{(j)} = x_{i+N+1} \quad if \quad x_{(j)} \geq x_{i-N}, \ \ x_{(j)} \leq x_{i+N+1}, \ \ and \ \ x_{(j+1)} > x_{i+N+1}.$$

Finally, the case shown in Fig.2c is

$$x'_{(j)} = x_{i+N+1} \quad if \quad x_{(j)} \leq x_{i-N}, \ \ x_{(j)} > x_i + N + 1, \ \ and \ \ x_{(j-1)} > x_{i+N+1}.$$

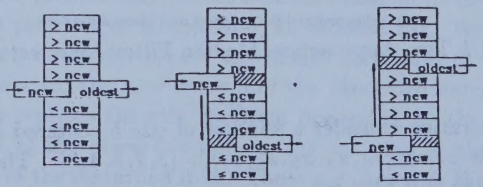


Fig.2 Possible Conditions for Updating the Ordered List.

Going back to the previous example, if $(x_3, x_4, x_5, x_6, x_7, x_8) = (2,2,5,4,3,4)$, then when the window is centered at i=5 we have

$$\mathbf{X}_5 = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{pmatrix} \quad and \quad \mathbf{R}_5 = \begin{pmatrix} (5)_{mod\ 5} = 5 \\ (6)_{mod\ 5} = 1 \\ (7)_{mod\ 5} = 2 \\ (3)_{mod\ 5} = 3 \\ (4)_{mod\ 5} = 4 \end{pmatrix}$$

If we shift the window to the next position we have

$$\mathbf{X}_6 = \begin{pmatrix} 5 \\ 4 \\ 4 \\ 3 \\ 2 \end{pmatrix} \quad and \quad \mathbf{R}_6 = \begin{pmatrix} (5)_{mod\ 5} = 5 \\ (6)_{mod\ 5} = 1 \\ (8)_{mod\ 5} = 3 \\ (7)_{mod\ 5} = 2 \\ (4)_{mod\ 5} = 4 \end{pmatrix}$$

A local control unit associated with each register determines the source of the next value-sequence number pair. This local control needs as input: 1) the value comparisons from its associated register, the one above and the one below, and 2) signals indicating whether this register is at, above or below the sample being discarded. The only non-local signal required is that indicating whether the register is above or below that containing the sample being discarded (assuming that it doesn't contain the sample to be discarded). In our system this comes from a "carry chain" that propagates a signal of one sign above the sample being discarded and of the other sign below.

The filter operates sequentially. The new sample value and its sequence number (mod 2N+1) are broadcast to the two sets of register-comparators (one for sample values, one for sequence numbers) and the comparison of the new values and those in each register pair are made locally. When the comparisons have been made and the signal has propagated on the "above/below" chain, the registers take on their new values. Then, the value register corresponding to the desired rank is outputted. The filter can be made recursive by, at this time in the sequence, replacing the value having a sequence number corresponding to the present output sequence number with the output sample value. The definition of the register vector, the input sequence vector and the updating process must now be modified in order to account for the recursive operation. In this case,

$$\mathbf{Z}_i = \begin{pmatrix} z_{(2N+1)} \\ z_{(2N)} \\ \vdots \\ z_{(1)} \end{pmatrix} \quad , and \quad \mathbf{R}_i = \begin{pmatrix} \alpha_{2N+1} \\ \alpha_{2N} \\ \vdots \\ \alpha_1 \end{pmatrix}$$

where $z_{(j)}$ is the jth largest sample among $y_{i-N}, ..., y_{i-1}, x_i, ..., x_{i+N}$, and where α_j is the number $(l)_{mod\ 2N+1}$ associated with the jth largest value such that either $x_l = z_{(j)}$ or $y_l = z_{(j)}$. Before we shift the window to the next position, the register and sequence vectors must go through the recursive stage as mentioned before $\mathbf{Z}_i' = \mathbf{Z}_i$ and all the elements of $\mathbf{R}_i$ are the same in $\mathbf{R}_i'$ except α_N which is $(i)_{mod\ 2N+1}$. The second stage is the updating of the register vector $\mathbf{R}_i'$ when y_{i-N} is discarded and x_{i+N+1} is added to the register. The procedure is the same as the one described for the non-recursive filter.

The size of the sample window is determined by the modulo of the sequence number counter and can be made programmable, within the limits imposed by the maximum number of registers available. The rank order output can also be made programmable. There are many ways to start the filter, the simplest being to load all value registers with zero or some other standard value and to load the set of sequence registers with all allowed sequence numbers. In our design, the value registers, sequence registers, associated comparators, and local control are all the same except for the provision for automatically loading a different number in each sequence register on startup.

III. IMPLEMENTATION

Our median filter design is built largely from the repetition of two quite similar

one bit register/compare cells, one for data and the other for sequence numbers. These cells are placed in a rectangular array forming two register sets as shown schematically in fig [3]. Each cell stores one bit of data or sequence number and provides one bit of the associated circuit for comparing the stored number with the number (data or sequence number) being broadcast. It also contains a multiplexor that selects the next bit to be stored from: the current contents of the register, the current contents of the corresponding bit in the storage register above or below, or from the broadcast data bus. Each of these one bit store and compare cells contains twenty transistors.

The two basic register/compare cells differ only in the kind of comparison they make between the new data and the data currently in the register and in the way they are initialized on startup. The comparator in each cell of the data register is part of a compare chain that determines whether the new data on the bus is greater than, or less than the data currently stored in the register. The equal condition can be associated with either greater than or less than with minor changes in the local next value logic. The comparator associated with the sequence number register tests for equality.

In our design, the comparator in each cell is part of a compare chain. For the data comparison, the compare signal must propagate from the low order bit to the high order bit. For the compare on equal circuit in the sequence number register, the compare signal may be arranged to propagate in either direction.

The decision as to which of the four possible values a given register will get next is made by a local control unit that has four inputs: 1) the result of the value comparison between the new value and the value presently stored in that register, 2) the result of the value comparison with the register above in the sequence, 3) the result of the value comparison with the register below, and 4) a signal which indicates where the given register is relative to the register containing the data to be discarded, that is the register pair where the sequence number (mod(2N+1)) equals the new sequence number.

This last input is the only information that is not local in the sense that it is derived from comparisons of the new data with the data stored in a given register

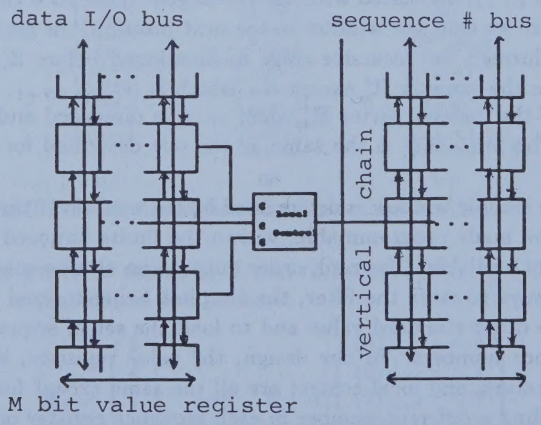


Fig.3 Internal Structure of the Ranked and Input Sequence Vector Network.

or its immediate neighbors. In our design, this "above/below the discard" signal is provided by a "vertical" chain running through the register set. The signal on this chain has one value above the register containing the data being replaced and the other value below. This signal is derived from the propagation along a compare chain in the sequence number register followed by propagation along the vertical chain. It is possible to overlap this step with the outputting of the last median computed. Our simulations show that the worse case propagation delay along these two chains (which occurs in our case when the top register set contains the value being discarded) does not limit the ultimate speed of the filter.

The filter circuit operates sequentially. First, the new data and the sequence number (mod (2N+1)) are broadcast to the comparators. (The sequence number may be broadcast during the end of the preceding cycle, if necessary, to speed up the propagation of the result of the sequence number comparison.) When the output from the multiplexors on each register cell are stable, the new values are clocked into each register and the ordered list is updated. To reduce the size of the cell, our present design uses a scheme where the static register cell is momentarily separated into two dynamic register cells, one retaining the old value and the other being updated. The static flip flop is reconnected in a sequence that assures that the new value is retained. This scheme avoids any race condition in the registers during update and uses a minimum of parts at the cost of some clocking complexity and resultant loss in speed.

Finally, the median, or other selected output, is gated to the data bus. In our design, we output on the same internal bus that we use for broadcasting new values to the comparison circuits. This is of value in the recursive modification of the filter as described below, but for the non-recursive filter is slower than a scheme using separate input and output data busses.

There are a number of possible schemes for initializing the filter. The simplest, and the one which appeals to us, involves loading zero values (or the lowest possible value in a system allowing negative data) into each of the data registers and successive numbers (starting from the top) into the sequence value registers. With this scheme it is very easy to program the size of the filter to any size less or equal to that of the number of register pairs in the filter by simply controlling the modulo number of the counter generating the sequence numbers. If the counter is programmed to be mod(M), the sequence value registers beyond the Mth are pre-programmed with a sequence number never encountered and the associated data registers contain the smallest possible data value and are thus also inactive.

Each register set also has a decoder which is used to select the register whose value is to be outputted. If the register selected is in the middle of the M active registers, we have a median filter. If some other register is selected we have the corresponding rank filter.

The filter can be made recursive by broadcasting the sequence number corresponding to the output value while the output is on the data bus and activating the previously described mechanism for inserting values in sequence into the register set using the output value as input data.

IV. SIZE AND SPEED CONSIDERATIONS

The chip,with the architecture of the median filter presented here, has been designed to be implemented with an NMOS process with $\lambda = 2\ mum$ (feature size = $4\ mum$).

The area of the chip can be represented as:

$$Area = A + F\,m + B\,n\,[\,C\,(m + \alpha\,(n)) + D\,] \quad (1)$$

where

$\alpha(n) = integer\ (\ log_2\ (n-1) + 1\ )$

n: window size

m: number of input bits

A: area of, internal clock, clock inputs, control inputs, sequential register

F: area of, input/output buffer, line driver, (per input bit)

B: height of the block (height of the tallest element)

C: width of the value register and sequential register

D: width of rank latch,end terminations,local control and carry chain

$\alpha(n)$ is the number of bits needed by the sequential register as function of the window size.

From equation (1) we can see that the area is composed of two constant terms A and F times m; plus the third term which express the increase in area as a function of the maximum window size. Then equation (1) can be expressed as

$$Area = A + F\,m + T(n) \quad (2)$$

where

$$T(n) = B\,n\,[\,C\,(m + \beta\,(n) + D\,] \quad (3)$$

It can be shown that D is approximately 5C , and if we assume that the input sequence has eight bit resolution, then equation (3) becomes

$$T(n) = B\,C\,n\,(13 + \beta\,(n)\,) \quad (4)$$

In order to compare this implementation with others in the literature, e.g. the one presented by Oflazer [7], we compare the expression of equation (4) with one that grows as the square of the window size. To make a reasonable comparison, we must compare the areas of both implementations for the same window size. Table 1 shows values for the different areas, expressed in mm^2 , $\lambda = 2\ \mu m$,$B = 100\ \lambda$, and$C$=70λ. It must be pointed out that in the implementation presented here, the area ($A + Fm$), has been kept constant when changing the window size from 15 to 5. Also the values used for the Oflazer implementation are estimated from his paper, assuming an active sorting area of 50 percent of the chip. These curves are presented in Fig. 4. From this figure we can conclude that, taking in account the assumptions made here, both implementations are comparable in size up to a window size of about 7, for small window sizes, the chip area is dominated by the constant terms, and that for large window sizes the Sorted-List Implementation is much smaller.

AREA in mm^2 vs. n				
	n	$A+Fm$	$T(n)$	Area
Udel	15	20	8	28
work	5	20	3	23
Oflazer[4]	5	10	10	20

Table 1. Area vs. Window Size

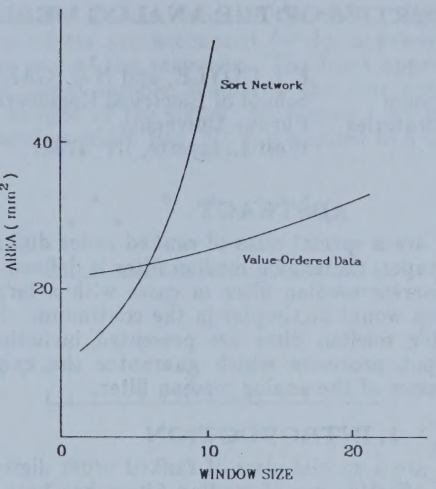


Fig.4 Area vs. Window Size.

We have simulated the performance of the circuit described. With arbitrary clock precision our calculations predict performance at a rate exceeding 20 megasamples per second. With realistic constraints on clock timing, the filter as designed should operate at real time with 60hz video signals on a 256 by 256 raster and possibly with 30hz video on a 512 by 512 raster.

V. CONCLUSIONS

We presented an architecture for median filtering operations that is well suited for parallel computation and local control. The window size and the rank output can be easily modified making this implementation flexible for use in many signal processing applications. The hardware complexity is slightly increased as the window size is increased and the speed of this chip is fast enough for real time video processing.

VI. REFERENCES

[1] T.S. Huang, ed. , *Two Dimensional Signal Processing II* . Topics in Applied Physics, Vol. 43. Springer-Verlag, Berlin 1981.

[2] G.R. Arce , N.C. Gallagher and T.A. Nodes, *Median Filters: Theory for One and Two Dimensional Filters - Advances in Computer Vision and Image Processing* JAE Press, (to be published 1985).

[3] T.A. Nodes and N.C. Gallagher. " Median Filters: Some Modifications and Their Properties," *IEEE Trans. on ASSP*, Vol. ASSP-30, no. 5, October 1982.

[4] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, MA: Addison Wesley, 1974.

[5] T.S. Huang, G.J. Yang, and G.T. Tang, "A Fast Two-Dimensional Median Filter Algorithm , " *IEEE Trans. on ASSP*, Vol. ASSP-27, no. 1, October 1979.

[6] E. Ataman, V.K. Aatre, and K.M. Wong, " A Fast Method for Real Time Median Filtering ," *IEEE Trans. on ASSP*, Vol. ASSP-28, no. 4, August 1980.

[7] K.Oflazer , " Design and Implementation of a Single Chip 1-D Median Filter ," *IEEE Trans. on ASSP*, Vol. ASSP-31, no. 5, October 1983.

STATISTICAL PROPERTIES OF THE ANALOG MEDIAN FILTER

J.P. FITCH
Engineering Research Division
Lawrence Livermore Laboratories
Livermore, CA 94550

E.J. COYLE, and N.C. GALLAGHER
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

Discrete median filters are a special class of ranked order digital filters used for smoothing signals. In this paper, the analog median filter is defined and proposed for analysis of the standard discrete median filter in cases with a large sample size or when the associated statistics would be simpler in the continuum. Several stochastic interpretations of the analog median filter are presented including necessary and sufficient conditions on input processes which guarantee the existence of output distributions for multiple passes of the analog median filter.

1. INTRODUCTION

Discrete median filters are a special class of ranked order digital filters used for smoothing signals [1]. One effective use of median filters has been the reduction of high frequency and impulsive noise in digital images without the extensive blurring and edge destruction associated with linear filters [2]. Other applications include the smoothing of noisy pitch contours in speech signals and data compression using the root signal properties combined with a block truncation coding (BTC) technique [3,4]. The use of the median filter is not prompted by some known analytical optimality condition; instead, its use is justified by the observed success of the filter in the previously mentioned areas.

When random signals are used as inputs, the nonlinear nature of the median filter makes the output distributions difficult to calculate and to comprehend [5]. The threshold decomposition technique [6] for reducing multi-level signals to binary signals does not reduce the statistical computation involved. We need a simpler means of calculating or approximating the output distributions of median filters. The basic idea behind the threshold decomposition--that of sliding a threshold through the various levels of a signal, lends itself to extensions in the domain of many leveled signals. In fact we might approach this problem by letting the number of allowable levels and the support of the input functions go to the continuum. We call the filter associated with the median operator, where these input functions are allowed, the analog median filter. In many cases this extension will make the analysis simpler than for the standard median filter. Because the definition is based on the threshold decomposition technique, however, the same underlying structure is maintained in both filters.

In this paper, we define and present a preliminary analysis of the analog median filter. The next section introduces the analog filter as the limiting case of the discrete median filter. In Section 3, a random variable interpretation of the analog filter is given. Necessary and sufficient conditions on stochastic input processes which guarantee the existence of output distributions for multiple filter passes are presented in Section 4. Examples and Conclusions follow in the final sections of the paper.

2. ANALOG MEDIAN FILTER DEFINITION

The implementation of a standard discrete median filter requires a simple nonlinear digital operation. To begin, take a sampled signal of length L; across this signal slide a window that spans $2N+1$ points. The filter output at each window position is given the same position as the sample point at the center of the window

This work was supported by the National Science Foundation under grant ECS83-6235.

and is set equal to the median value of the $2N+1$ signal samples in the window. Start up and end effects are accounted for by appending N samples to both the beginning and the end of the sequence. The front appended samples are given the value of the first signal sample; similarly, the rear appended samples receive the value of the last sample of the signal. In Figure 1 we present an example where a median filter of window width five, $N=2$, is applied to a signal with three levels.

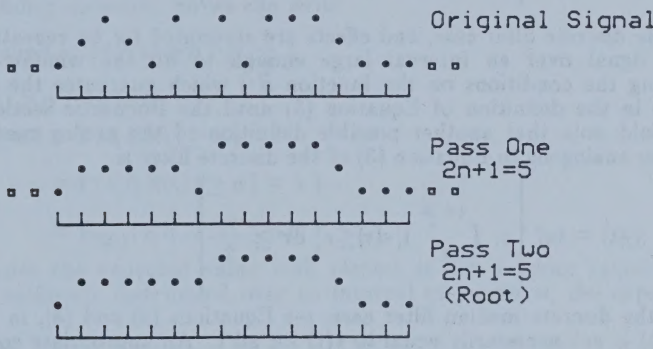


Figure 1: Discrete median filter example.

The output is given for each pass of the median filter until a root signal is reached. Recall that the basic idea is to rank the samples in the window and select the median value as the filter output.

We assume throughout this section that the input to the discrete median filter is a sequence of length L which takes on the value a(m) at position m, $1 \leq m \leq L$. Let b(m) be the output at position m of a standard median filter with window width $2N+1$ applied to the input sequence a(m). Then by definition

$$b(m) = \text{median}\{\, a(m-N),\ldots,a(m),\ldots,a(m+N)\,\} \tag{1}$$

$$= \max\{\, i\text{: At least } N+1 \text{ of } a(m-N),\ldots,a(m+N) \text{ are} \geq i\,\}$$

$$= \max\{\, i\text{: More than half of } a(m-N),\ldots,a(m+N) \text{ are} \geq i\,\} \tag{2}$$

In a similar manner we might have shown

$$b(m) = \min\{\, i\text{: More than half of } a(m-N),\ldots,a(m+N) \text{ are} \leq i\,\} \tag{3}$$

Equations (2) and (3) are equivalent because there is an odd and finite number of values in the ranking window. Later we will see that these equations are equivalent only when the median is unique.

In the definition of the analog median filter, and in other definitions, discussions, and proofs that follow, it is convenient to use the indicator function of the event A which we define as

$$I(A) = \begin{cases} 1 & \text{,if A is true} \\ 0 & \text{,else} \end{cases} \tag{4}$$

Also, the notation $\sup(\cdot)$ and $\inf(\cdot)$ will be used to denote the supremum and infimum operations, respectively.

For any appropriate definition, the analog filter that we propose should be a limiting case to the discrete filter. The following definition has this property and includes discrete median filters as a special class of input functions [7].

Definition: We say that $y(\cdot)$ is the output of a *analog median filter* of window width w applied to an input signal $s(\cdot)$ on the interval [c,d) if for almost every $t \in [c,d)$

$$y(t) = \sup\{\, r\text{: More than half of } s(\tau), \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \text{ are} \geq r \,\}$$

$$= \sup\left\{ r\text{: } \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[s(\tau) \geq r]\, d\tau \geq \frac{w}{2} \right\} \tag{5}$$

As in the discrete filter case, end effects are accounted for by repeating the endpoint of the signal over an interval large enough to fill the window [1]. We delay discussing the conditions on the function $s(\cdot)$ which guarantee the existence of the integral in the definition of Equation (5) until the stochastic section of the paper. We should note that another possible definition of the analog median filter which would be analogous to Equation (3) of the discrete filter is

$$y_l(t) = \inf\left\{ r\text{: } \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[s(\tau) \leq r]\, d\tau \geq \frac{w}{2} \right\} \tag{6}$$

Unlike the discrete median filter case, see Equations (2) and (3), in the analog filter case $y_l(t)$ is not necessarily equal to $y(t)$ for all t. An appropriate compromise might be to take the average of $y_l(t)$ and $y(t)$ to obtain the output of the analog median filter. This additional computation seems cumbersome and usually unnecessary; we therefore resolve to define the output of the analog median filter by Equation (5). Hopefully this definition will allow us to go back and forth between discrete and analog signals so that results which are easier to obtain in one domain can be transferred to the other domain.

There are other equivalent definitions for the analog median filter. As an example of an alternative definition, assume that $m(\cdot)$ is a measure, such as the Lebesgue measure, defined for sets of real numbers. If the measure of an interval is its length and the measure of the sum of disjoint sets is the sum of the measures of each set we could use the following as the definition of a window width w analog median filter applied to the input signal $s(\cdot)$.

$$y(t) = \sup\left\{ r\text{: } m\left[\tau\text{: } s(\tau) \geq r \text{ and } t-\frac{w}{2} \leq \tau < t+\frac{w}{2} \right] \geq \frac{w}{2} \right\} \tag{7}$$

End effects are accounted for as in the previous definition. This measure theoretic definition can be easier to manipulate than the integral definition of Equation (5). The proofs in the stochastic section of the paper demonstrate the utility of this definition.

3. RANDOM VARIABLE INTERPRETATION OF THE FILTER

The purpose of this section of the paper is to describe the various random and stochastic interpretations of the analog median filter. We begin by showing that an analog median filter applied to a deterministic function $f(\cdot)$ at point t is equivalent to calculating the median of the distribution of a random variable $Z(t)=f(X(t))$, where $X(t)$ is uniformly distributed on $[t-\frac{w}{2}, t+\frac{w}{2})$. To obtain the entire output of the filter we allow t to range over the support of $f(\cdot)$. The final random interpretation, which is presented in the next section, is for the case of a stochastic process as the input to the analog median filter. Necessary and sufficient conditions for the existence of filter output distributions are discussed.

Theorem 1: The output $y(t)$ of an analog median filter applied to a measurable function $f(\cdot)$ equals the median of the distribution of the random variable $Z(t)=f(X(t))$, where $X(t)$ is uniformly distributed on $[t-\frac{w}{2}, t+\frac{w}{2})$.

Proof: First note that the median of a distribution is not always uniquely defined. We avoid discrepancies with our filter definition by always using the largest value α such that

$$P(Z(t) \leq \alpha) = P(Z(t) \geq \alpha) \geq \frac{1}{2} \tag{8}$$

as the median of the distribution which we denote as $\alpha_0(t)$. Note that $P(\cdot)$ is the associated probability measure. So we can write

$$\alpha_0(t) = \sup \{ \alpha: P(Z(t) \geq \alpha) \geq \frac{1}{2} \} \tag{9}$$

Expanding $P(Z(t) \geq \alpha)$ we have

$$\begin{aligned} P(Z(t) \geq \alpha) &= P(f(X(t)) \geq \alpha) \\ &= P(I[f(X(t)) \geq \alpha] = 1) \\ &= E_{X(t)}(I[f(X(t)) \geq \alpha]) \end{aligned} \tag{10}$$

where $E_{X(t)}$ denotes the expected value with respect to the random variable X(t). Because X(t) is uniformly distributed over an interval of length w, the expectation reduces to

$$P(Z(t) \geq \alpha) = \frac{1}{w} \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[f(x) \geq \alpha] \, dx \tag{11}$$

Combining the results of Equations (9) and (11) we have

$$\alpha_0(t) = \sup \{ \alpha: \frac{1}{w} \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[f(x) \geq \alpha] \, dx \geq \frac{1}{2} \} \tag{12}$$

$$= \sup \{ \alpha: \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[f(x) \geq \alpha] \, dx \geq \frac{w}{2} \}$$

$$= y(t)$$

where y(t) is the output at position t of an analog median filter applied to a deterministic function $f(\cdot)$.

□

4. RESPONSE TO STOCHASTIC INPUTS

In the previous section, Theorem 1 provided an intuitive interpretation to the nature of the analog median transformation operating on deterministic functions. The usefulness of this interpretation depends on one's familiarity with random variables. Assuming a familiarity, the next step would be to allow stochastic inputs to the filter. In order to proceed in our analysis of random inputs we must first define what is meant by a stochastic process and then determine what class of processes are allowable as inputs to the filter. The following definitions and theorems on stochastic processes are from Doob [8].

We define a stochastic process as any family of random variables $\{x(t,\omega), t \in T\}$. For any fixed $t \in T$, $x(t,\omega)$ is a random variable and is therefore measurable with respect to ω. For a fixed ω, we call the resulting function of t a sample function of the process. Applying a window width w analog median filter to the stochastic input $x(t,\omega)$ results in $y(t,\omega)$ where

$$y(t,\omega) = \sup\left\{ r: \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[x(\tau,\omega) \geq r]\, d\tau \geq \frac{w}{2} \right\}$$

$$= \sup\left\{ r: m_\ell\{ \tau: x(\tau,\omega) \geq r,\ \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \} \geq \frac{w}{2} \right\} \tag{13}$$

Where the subscript ℓ on m_ℓ denotes Lebesgue measure and is used to distinguish it from the measure $m(\cdot)$ associated with the probability space. In the definition of a stochastic process no restrictions had been placed on the "t" properties of the function. Our first restriction is to set T equal to some finite interval. From Equation (13) it is obvious that in order for the analog median filter to make sense the process must be measurable with respect to the set of all half open intervals of length w with midpoint in the interval T. A process satisfying this condition is called measurable. Specifically, the stochastic process $\{x(t,\omega), t \in T\}$ is called *measurable* if the parameter set T is Lebesgue measurable and if $x(t,\omega)$ defines a function measurable in the pair of variables (t,ω). The importance of the measurability of the input stochastic process is demonstrated by the following Property.

Property 1: The output distribution of an analog median filter applied to stochastic process $x(t,\omega)$ exists if and only if $x(t,\omega)$ is a measurable process.

Proof: Let $y(t,\omega)$ denote the output of the window width w filter. Then

$$y(t,\omega) = \sup\left\{ r: m_\ell\{ \tau: x(\tau,\omega) \geq r,\ \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \} \geq \frac{w}{2} \right\} \tag{14}$$

For convenience let $A(r,t,\omega)$ be the set of all τ in the interval $[t-\frac{w}{2}, t+\frac{w}{2})$ such that $x(t,\omega)$ is greater than or equal to r. That is,

$$A(r,t,\omega) = \{ \tau: x(\tau,\omega) \geq r, \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \} \tag{15}$$

We should note that $A(r,t,\omega)$ is a Lebesgue measurable set for any t or r if and only if $x(t,\omega)$ is a measurable process. Substituting this into Equation (14)

$$y(t,\omega) = \sup\{ r: m_\ell[A(r,t,\omega)] \geq \frac{w}{2} \} \tag{16}$$

The proof is easily done by investigating the function $G(\cdot)$ where $G(\gamma)$ equals the probability that $y(t,\omega)$ is greater than or equal to γ. If $G(\gamma)$ exists we have

$$G(\gamma) = P(y(t,\omega) \geq \gamma)$$

$$= P(\sup[r: m_\ell\{A(r,t,\omega)\} \geq \frac{w}{2}] \geq \gamma) \tag{17}$$

By noting that $A(r,t,\omega)$ is a subset of $A(\gamma,t,\omega)$ for any $r \geq \gamma$ and using the order of the inequalities, we can eliminate the supremum from Equation (17).

$$G(\gamma) = P(m_\ell\{A(\gamma,t,\omega)\} \geq \frac{w}{2})$$

$$= m(m_\ell\{A(\gamma,t,\omega)\} \geq \frac{w}{2})$$

$$= m(m_\ell\{ \tau: x(\tau,\omega) \geq \gamma, \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \} \geq \frac{w}{2}) \tag{18}$$

This set of equalities imply that $G(\gamma)$ exists for all γ if and only if $x(t,\omega)$ is measurable with respect to the Lebesgue measure $m_\ell(\cdot)$ and the probability measure $m(\cdot)$. Note that $G(\gamma)$ and consequently the distribution function $F(\gamma)$ may exist for certain real γ without $x(t,\omega)$ being a measurable process. However, the definition of a distribution function $F(\cdot)$ requires $F(\gamma)$ to equal $P[x(t,\omega) \leq \gamma]$ for any number γ from $-\infty$ ∞. Combining this definition with the properties of $G(\cdot)$ results in the necessary and sufficient existence conditions of the distribution function.
□

This Property establishes the necessary and sufficient conditions on the input process which guarantee the existence of the output distribution. We would like to remain in the class of measurable processes even for repeated passes of the analog filter. By the previous Property, this would guarantee output distributions for any countable number of filter passes. To do this we first show that

Property 2: The stochastic output of an analog median filter is measurable with respect to its parameter space if and only if the input process is measurable.

Proof: Let $y(t,\omega)$ denote the output of a window width w analog median filter applied to a stochastic process $x(t,\omega)$. To show that the output is a measurable function in t, it is sufficient to show that $\Gamma = \{t: y(t,\omega) \geq \gamma\}$ is a measurable set for any real γ. Expanding Γ we have

$$\Gamma = \{t: y(t,\omega) \geq \gamma\}$$

$$= \left\{ t: \sup \left[r: m_\ell \{ \tau: x(\tau,\omega) \geq r, \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \} \geq \frac{w}{2} \right] \geq \gamma \right\}$$

Because of the order of the inequalities we can eliminate the supremum in a manner similar to the proof of Property 1.

$$\Gamma = \left\{ t: m_\ell \{ \tau: x(\tau,\omega) \geq \gamma, \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \} \geq \frac{w}{2} \right\}$$

$$= \{t: B(t) \geq \frac{w}{2}\} \tag{19}$$

where

$$B(t) = m_\ell \{ \tau: x(\tau,\omega) \geq \gamma, \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \}$$

Equation (19) has reduced the problem to showing that $B(t)$ is a measurable function. It is in fact easier to show the stricter condition that $B(t)$ is continuous in t. For any $\epsilon > 0$, we have

$$B(t+\epsilon) = m_\ell \{ \tau: x(\tau,\omega) \geq \gamma, \tau \in [t+\epsilon-\frac{w}{2}, t+\epsilon+\frac{w}{2}) \}$$

$$= m_\ell \{ \tau: x(\tau,\omega) \geq \gamma, \tau \in [t+\frac{w}{2}, t+\epsilon+\frac{w}{2}) \cup [t-\frac{w}{2}, t+\frac{w}{2}) - [t-\frac{w}{2}, t+\epsilon-\frac{w}{2}) \}$$

By the properties of the Lebesgue measure this equals

$$B(t+\epsilon) = m_\ell \{ \tau: x(\tau,\omega) \geq \gamma, \tau \in [t+\frac{w}{2}, t+\epsilon+\frac{w}{2}) \}$$

$$+ B(t) - m_\ell \{ \tau: x(\tau,\omega) \geq \gamma, \tau \in [t-\frac{w}{2}, t+\epsilon-\frac{w}{2}) \}$$

Because the Lebesgue measure of an interval is its length we have the following set of inequalities.

$$B(t+\epsilon) \geq B(t) - \epsilon$$
$$B(t+\epsilon) \leq B(t) + \epsilon$$

Which implies that $|B(t+\epsilon)-B(t)| \leq \epsilon$ for any $\epsilon>0$. Therefore B(t) is continuous and necessarily a measurable function.
□
Combining Properties 1 and 2 we have the following
Theorem 2: The output of an analog median filter is a measurable stochastic process if and only if the input process is measurable.

Property 1 implies that in order to have the existence of output distributions of the analog filter we must have a measurable input process. By the above Theorem, if we start with a measurable input process, we will always obtain measurable processes as output. This allows multiple filter passes to be performed without destroying the measurability of the process.

5. EXAMPLES AND DISCUSSION

The properties of the Dirac delta function make it a popular test input for linear systems. In fact, linear systems are often characterized by their response to this impulsive input. One property of the analog median filter is that its impulse response does not completely characterize its performance. We do the analysis for the purpose of comparison with the responses of the more common class of linear filters. Let $\delta(t)$ denote the Dirac delta function which is defined by the two properties $\int_{-\infty}^{\infty} \delta(t)\,dt = 1$ and $\delta(t)$ identically zero outside any ϵ neighborhood of $t=0$.
In anticipation of the result, we might note that the discrete median filter performs well in the elimination of additive impulsive noise. The notion that the analog median filter should have no response to a delta function is stated and proven in the following
Property 3: The response of a window width w analog median filter to a Dirac delta input is identically zero.
Proof: Using the theory of generalised functions [9], we represent the delta function as the limit of a series of functions which integrate to one and converge to have both desired properties. For this proof the simplest of these functions is

$$\chi_n(t) = \begin{cases} n & ,\text{if } \dfrac{-1}{2n} < t < \dfrac{1}{2n} \\ 0 & ,\text{else} \end{cases} \tag{20}$$

Obviously the two defining properties of the Dirac delta hold because for any n

$$\int_{-\infty}^{\infty} \chi_n(t)\,dt = 1 \text{ and } \delta(t) = \lim_{n\to\infty} \chi_n(t) = \begin{cases} 0 & ,t \neq 0 \\ \infty & ,t = 0 \end{cases}$$

Applying a analog median filter with window width w to $\chi_n(t)$ we obtain $y_n(t)$ where

$$y_n(t) = \sup\left\{ r: \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[\chi_n(t)(\tau) \geq r]\,d\tau \geq \frac{w}{2} \right\}$$

If $w \leq 2/n$ then

$$y_n(t) = \begin{cases} n & ,\text{if } \dfrac{-1}{2n} < t < \dfrac{1}{2n} \\ 0 & ,\text{else} \end{cases}$$

and if $w > 2/n$ then $y_n(t)$ equals zero for all t. Obviously for any $w>0$ there exists an $N>0$ such that for all $n \geq N$ we have $w > 2/n$. This implies that $y_n(t)$ is zero for

all $n \geq N$ and for this representation of the delta function we have an identically zero output of the analog median filter.
□

It is not surprising that the analog median filter does not respond to impulsive inputs. The main purpose of demonstrating the preceding property is to show that this linear technique is ineffective in the total characterization of the analog median filter.

As another example of how classic linear techniques fail to completely characterize the median filter, we examine the response of the filter to sinusoidal inputs. Begin by fixing w, the window width, and call the output of the analog filter y(t). Let $s(\cdot)$ be the sinusoidal input of the form $s(t) = \sin(2\pi t/\lambda)$, where λ denotes the wavelength of the test signal.

For linear systems, the superposition property allows the filtering of signals which are comprised of sums of simple signals to be done by filtering the individual simple signals and then recombining using addition to obtain the output. So for this example, we would calculate the effect of filtering a sinusoid of wavelength λ and then let λ range over the class of wavelengths necessary to represent the input signal using Fourier techniques. As has been noted before in [6], linear techniques cannot be used to obtain the set of simple signals allowing an additive superposition operator. Now we use the analog median filter to demonstrate why frequency representations of the response of the median filter are difficult to interpret. We begin with three restrictions on λ and t which make the calculations less tedious.
Case 1: $t = \lambda k/2$, $k = 0, \pm 1, \pm 2, \ldots$ By the symmetry of $s(\cdot)$ we know that for this restricted set of time values we have $s(t+\tau) = -s(t-\tau)$. It is easy to show that the output of the analog median filter y(t) will be zero at these points. This set of points is precisely the set where s(t) is also zero. We conclude that for sinusoidal inputs, the zero crossings are invariant to analog median filtering.
Case 2: $w = k\lambda$, $k = 1,2,\ldots$ Whenever the window width w of the filter is an integer multiple of the wavelength of the sinusoidal input, a complete cycle of the input will always be in the window of the filter for any time position. That is, because $s(t) = \sin(2\pi tk/w)$ and by the periodicity of the sine function, we have for all t

$$y(t) = \sup\left\{ r: \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[\sin(2\pi k\tau/w) \geq r]\, d\tau \geq \frac{w}{2} \right\} = 0 \tag{21}$$

We conclude that the analog median filter has no response to sinusoids with $\lambda = w/k$. In terms of the frequency of the sinusoid we have $\nu = 1/\lambda = k/w$ as the class of frequencies with an identically zero response.
Case 3: $\lambda \geq w > 0$. This case is simple because we obtain a root after only one pass of the filter. In addition, the root is simply a "clipped" version of the input. This can be seen in Figure 2. Analytically we have

$$y(t) = \begin{cases} \sin\left[\pi\left(\frac{2k+1}{2} - \frac{w}{\lambda}\right)\right] & ,\left(\frac{2k+1}{4}\right)\lambda - \frac{w}{2} \leq t < \left(\frac{2k+1}{4}\right)\lambda + \frac{w}{2} \\ \sin[2\pi t/\lambda] & ,\text{else} \end{cases} \tag{22}$$

where $k = 0, \pm 1, \pm 2, \ldots$ From this we see that the filter is clipping the tops of the sinusoid to constant regions of width w/2. Therefore y(t) is a root to the analog median filter of window width w. This is a well behaved response in that the nonconstant monotonic regions are identical to the input sinusoid.

If we consider the transformation of the filter as being from a single frequency signal to some output signal y(t). Then we immediately note that, because of the constant regions, the transformation spreads the output over the entire spectrum. This means that the sinusoidal response curve, because the normalized output energy is plotted in terms of the input wavelength, may be a misleading description of the

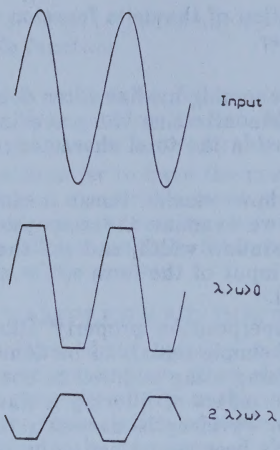


Figure 2: Analog median filter example.

filtering phenomenon. As an example, in Figure 3 we plot the response of the analog median filter to these sinusoidal signals in terms of the output energy normalized by the input energy.

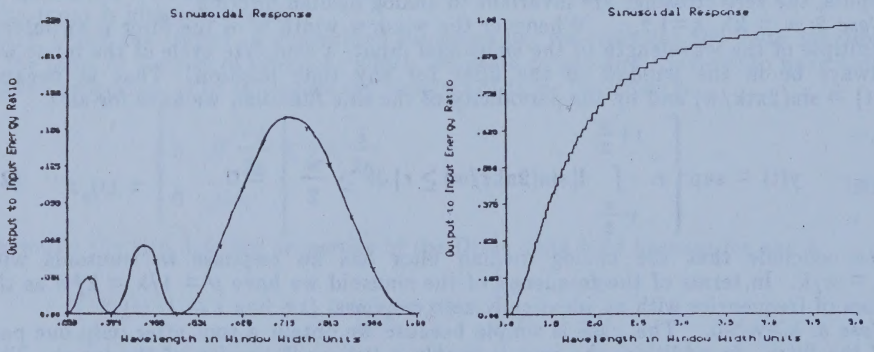


Figure 3: Sinusoidal response with $\frac{w}{4} \leq \lambda \leq 5w$.

It is important to remember that the output energy is spread over the spectrum even though it is the result of filtering an input which contains only one frequency.

This analysis has allowed us to calculate analytically the sinusoidal response of the filter for $\lambda \geq w > 0$. For smaller wavelengths the structure of the input sinusoid is destroyed making the analysis significantly more complicated and therefore it will not be presented here. Instead refer to the sinusoidal response plots in the figures. These plots were calculated using the approximation techniques presented earlier in this paper. The zeros and other predicted behavior from the previous three cases can be verified. Note that, as expected, as λ becomes much greater than w the energy ratio tends to one.

6. CONCLUSIONS

The major contribution of this paper is the definition of the analog median filter. The foundation of the theory necessary to use this filter to model discrete problems and to attack the difficult statistics involved with ranked operations is also given. This is a new concept for the analysis of median filters. Even though this paper is an introduction to the filter, its use as a tool for analytic proofs is demonstrated along with the foundation for statistical interpretations. The necessary and sufficient conditions for the existence of output distributions of multiple analog filter passes is an important result. The practical uses are demonstrated by the intuitive results which can be derived by treating the filter as an entity in itself and then mapping to the discrete domain for comparison with the already popular discrete median filter.

7. ACKNOWLEDGEMENTS

The authors wish to thank Leon J. Gleser, Professor of Statistics, Purdue University, for his helpful comments and discussions which resulted in the proof of Theorem 2.

8. REFERENCES

[1] N.C. Gallagher and G.L. Wise, "A theoretical analysis of the properties of median filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-29, pp. 1136-1141, Dec. 1981.

[2] T.S. Huang and G.J. Yang, "Median filters and their applications to image processing," School of Electrical Engineering, Purdue University, West Lafayette, IN; TR. EE 80-1 Jan. 1980.

[3] L.R. Rabiner, M.R. Sambur, and C.E. Schmidt, "Applications of a nonlinear smoothing algorithm to speech processing," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-23, pp. 552-557, Dec. 1975.

[4] G.A. Arce and N.C. Gallagher, Jr., "State description for the root-signal set of median filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-30, pp. 894-902, Dec. 1982.

[5] T.A. Nodes and N.C. Gallagher, "Theoretical results on the properties of median type filters," School of Electrical Engineering, Purdue University, West Lafayette, IN; TR-EE 82-32, Sept. 1982.

[6] J.P. Fitch, E.J. Coyle, and N.C. Gallagher, "Median filtering by threshold decomposition," *IEEE Trans. Acoust., Speech, Signal Processing,* to appear.

[7] J.P.Fitch, E.J. Coyle, and N.C. Gallagher, "The analog median filter," submitted to *IEEE Trans. Circuits and Systems.*

[8] J.L. Doob, *Stochastic Processes.* New York: John Wiley & Sons, 1953.

[9] M.J. Lighthill, *Introduction to Fourier Analysis and Generalised Functions.* Cambridge: University Press, 1958.

DECENTRALIZED CONTROL FOR A CLASS OF NONLINEAR SYSTEMS USING A VARIABLE STRUCTURE APPROACH

GREGORY P. MATTHEWS, STANISLAW H. ZAK, and RAYMOND A. DECARLO
School of Electrical Engineering, Purdue University, West Lafayette, Indiana

ABSTRACT

This paper develops a method of variable structure control (VSC) for a class of nonlinear interconnected systems. The control objective entails using switched feedback to make each subsystem behave as a controllable linear subsystem having a desired behavior. Two advantages result: the algorithm is amenable to decentralized pole placement techniques and permits extending single input VSC to the multi-input case. For a special subclass of the systems considered, an algorithm to compute the feedback gains is developed via the use of the Kronecker product. A simple nonlinear example illustrates the techniques and extensions discussed.

1. INTRODUCTION

Nonlinear and interconnected (large scale) systems go hand in hand. Various techniques exist for the linear case. Little, however, is available for the nonlinear case. Usually one linearizes a nonlinear system about an operating point. Unfortunately, linearization is valid only in a neighborhood (usually small) of the given operating point.

Decentralized variable structure control (DVSC) offers a viable alternative. By high speed switching, DVSC drives the state trajectory of each nonlinear subsystem to a special manifold (called the sliding surface) in R^n and maintains it there. This manifold represents the equivalent "linear" system designed a priori to have a desired system behavior.

The literature contains much on global VSC. DVSC represents an area of promising research with only minimal exposure in the literature: in [11] Morgan and Ozguner use VSC to control a robot manipulator without developing a linear equivalent system. In [5,6], a linear interconnection of single input nonlinear subsystems is considered. The control objective was to make the system behave as linear interconnected system with prescribed poles. In [3], Hawley and DeCarlo mold this technique to a simple tracking problem for an interconnection of two synchronous machines.

2. MODEL FORMULATION

The model formulation utilized in this paper is the *Component Connection Model* (CCM) [8]. The CCM represents the system as a series of N subsystem state models with the algebraic interconnections. The i-th subsystem is expressed as a m_i input n_i–th order state model as

$$\dot{x}_i(t) = f_i(x_i(t),t) + B_i(a_i(t) + u_i(t)) = A_i(x_i(t),t)\,x_i(t) + B_i(a_i(t) + u_i(t)) \quad (2.1)$$

$$b_i(t) = C_i x_i(t) \quad (2.2)$$

with linear interconnections given by

$$a_i(t) = \sum_{j=1}^{N} L_{i,j}\, b_j(t)\,. \quad (2.3)$$

For each instant of time $x_i \in R^{n_i}$, $u_i \in R^{m_i}$, $a_i \in R^{m_i}$, $b_i \in R^{p_i}$ are the i-th subsystem's state, feedback input, interconnection input, and output vectors respectively. Further, $L_{i,j} \in R^{m_i \times m_j}$, $B_i \in R^{n_i \times m_i}$, and $f_i(x_i,t) \in R^{n_i}$. In addition the

following assumptions are made: (1) every state x_i^j is locally measurable, (2) (A_i,B_i) is controllable in the large (CIL) [13], (3) B_i has only constant entries, (4) any nonlinear or time varying term in A_i is in the image of B_i.

The composite system state model takes the form

$$\begin{aligned} \dot{x}(t) &= F(x,t)x(t) + Bu(t) \\ b(t) &= Cx(t) \end{aligned} \tag{2.4}$$

where

$F = A + BLC$, $x = col[x_1, \cdots, x_N] \in R^n$, $n = \sum_{i=1}^{N} n_i$, $m = \sum_{i=1}^{N} m_i$, $p = \sum_{i=1}^{N} p_i$,

$u = col[u_1, \cdots, u_N] \in R^m$, $a = col[a_1, \cdots, a_N] \in R^m$, $b = col[b_1, \cdots, b_N] \in R^p$,

and $A=block-diag[A_1, \cdots, A_N]$, $B=block-diag[B_1, \cdots, B_N]$, etc.

By a decentralized control we mean that the control u_i for the i-th subsystem should be a function of the variables which can be measured at the i-th subsystem, the local states and the local subsystem inputs as determined by the interconnections with other systems and local feedback. The problem is now to determine a switched control $u_i = col[u_i^1, \cdots, u_i^{m_i}]$, such that the above system is driven to the sliding surface from any initial condition where the decentralized discontinuous [1] control takes the form

$$u_i^j = \begin{cases} u_i^{j,+} & \text{if } \sigma_i^j(x_i,a_i) > 0 \\ u_i^{j,-} & \text{if } \sigma_i^j(x_i,a_i) < 0 \end{cases} \tag{2.6}$$

where each $u_i^{j,\pm}$ is a continuous function of x,t.

Each local sliding surface is the intersection of m_i switching surfaces. We define the N sets of local switching manifolds or switching surfaces in R^n by $\sigma_i(x_i,a_i) = 0 \in R^{m_i}$, $i=1, \cdots, N$. With $\sigma_i(x_i,a_i)$ defined by

$$\sigma_i(x_i,a_i) = S_i x_i - \int_{-\infty}^{t} S_i B_i a_i(q)\, dq. \tag{2.7}$$

where $S_i \in R^{m_i x n_i} i=1, \cdots N$ are the matrices of containing the switching surface parameters. This matrix will be referred to as the switching surface matrix.

Our goal is to construct a set of local VSC controllers which will drive the state trajectory of each subsystem to and maintain it on the intersection of the m_i local switching surfaces; $\sigma_i^j = 0$ $j=1, \cdots, m_i$,$i=1,\ldots,N$. When this happens the system can be described by a lower order ($n_i - m_i$) set of equations. If the conditions (1)-(4) are satisfied, then this lower order system restricted to the intersection of the switching surfaces is linear and time invariant. By proper selection of the elements $[s_i^{j,k}]$ of the switching surface matrices one can ensure that once a solution to (2.1) lies on the sliding surface it converges to the origin, or an appropriate operating point, at the desired rate.

3. DESIGN PROCEDURE

The VSC design procedure consists of two primary phases. The first phase entails constructing each subsystem's switching surfaces so that the system restricted to the intersection of the switching surfaces has a desired behavior [1]. In the second phase of the DVSC design procedure, the control law is chosen so that a *sliding mode* exists on the intersection of these manifolds and this intersection can be reached from anywhere in the state space [1]. Specifically, phase two is the selection of a decentralized control law, such that the original system is Lyapunov stable to $\sigma(x) = 0$ for all states in the state space.

3.1 PHASE ONE: SELECTION OF THE SWITCHING SURFACES.

To select the elements $[s_i^{j,k}]$ $i=1,\cdots,N$ $j=1,\cdots,m_j$, $k=1,\cdots,n_j$, of (2.6) we use an extension of the method in [2] and make a series of local state transformations to coordinate systems in which the first $n_i - m_i$ states form the equivalent system when restricted to the sliding surface. This transforms the problem of selecting the entries of S_i into a decentralized eigenvalue placement problem. Once transformed the techniques of [7,9] may be used to assign a desired spectrum to the equivalent system and hence define the $[s_i^{j,k}]$.

For the i-th subsystem or component, make the local, linear, time-invariant, orthonormal state transformation $z_i = T_i x_i$ where

$$T_i = \begin{bmatrix} T_i^1 \\ T_i^2 \end{bmatrix}, \quad T_i^1 \in R^{(n_i-m_i)\times(n_i-m_i)}, \quad T_i^2 \in R^{m_i \times m_i} \tag{3.1a}$$

with

$$T_i^1 B_i = 0, \quad T_i^2 B_i = B_i^2 \tag{3.1b}$$

and

$$T_i A_i T_i^{-1} = \begin{bmatrix} A_i^{11} & A_i^{12} \\ A_i^{21} & A_i^{22} \end{bmatrix}. \tag{3.1c}$$

For the assumed system structure, these transformation matrices T_i, $i=1,\ldots,N$, simply permute the rows of B_i. A more general and numerically implementable method can be found. Observe that the rows of T_i^1 span the left null space of B_i, while the rows of T_i^2 span the image of B_i. A numerically efficient method of determining such matrices T_i, $i=1,\ldots,N$, the singular value decomposition (SVD) [12]. Using a SVD of B_i to obtain T_i always produces an orthonormal matrix T_i which satisfies (3.1).

With the above transformation of coordinates (2.1) becomes

$$\begin{aligned} \dot{z}_i^1 &= A_i^{11} z_i^1 + A_i^{12} z_i^2 \\ \dot{z}_i^2 &= A_i^{21} z_i + A_i^{22} z_i^2 + B_i^2 (u_i + a_i) \end{aligned} \tag{3.2}$$

where $z_i^1 \in R^{n_i - m_i}$, $z_i^2 \in R^{m_i}$, and B_i^2 is a $m_i \times m_i$ nonsingular matrix. The switching surface vectors then become

$$\sigma_i(z_i, a_i) = S_i^1 z_i^1 + S_i^2 z_i^2 - \int_{-\infty}^{t} S_i^2 B_i^2 a_i(q)\, dq. \tag{3.3}$$

where $[\, S_i^1 \mid S_i^2 \,] = S_i T^{-1}$. The composite switching surface vector is

$$\sigma(z) = S^1 z^1 + S^2 z^2 - \int_{-\infty}^{t} S B a(q)\, dq \tag{3.4}$$

where $S^i = \text{block-diag}\,[\, S_1^i, \cdots, S_N^i \,]$, $a = \text{col}\,[\, a_1, \cdots, a_N \,]$, $z^i = \text{col}\,[\, z_1^i, \cdots, z_N^i \,]$ $i=1,2$.

To insure a linear equivalent system, the switching surfaces must be linear in z. Since the first two terms are obviously linear, only the linearity of the third term must be insured. This requires that

$$\int_{-\infty}^{t} S B a(q)\, dq = G z^1, \quad G \in R^{m \times (n-m)} \tag{3.5}$$

For suitable G. This is always possible with proper insertion of dynamic compensation assuming B is a constant matrix.

Let $\int_{-\infty}^{t} S_i B_i a_i(q)\, dq = \tilde{a}_i(t)$ where $\tilde{a} = \text{col}[\tilde{a}_1, \ldots, \tilde{a}_N] = G z^1$. Hence equation (3.3) becomes

$$\sigma_i(z_i, \tilde{a}_i) = S_i^1 z_i^1 + S_i^2 z_i^2 - \tilde{a}_i = 0 \tag{3.6}$$

Solving for z_i^2 in terms of z_i^1, selecting $S_i^2 = (B_i^2)^{-1}$, and inserting into (3.2), yields

$$\dot{z}_i^1 = (A_i^{11} - A_i^{12} B_i^2 S_i^1) z_i^1 + A_i^{12} B_i^2 \tilde{a}_i \quad i=1,2,...,N . \tag{3.7}$$

The equivalent composite system becomes

$$\dot{z}^1 = [A^{11} - A^{12} B^2 S^1] z^1 + A^{12} B^2 G z^1 = \hat{A}_{eq} z^1 \tag{3.8}$$

If A_i^{11}, A_i^{12}, B_i^2, and G are constant matrices, and (A_i^{11}, A_i^{12}) is controllable, i=1,...,N, then using the method of [7] we may arbitrarily set the spectrum of $\hat{A}_{eq}$.

The following theorems given without proof assure us that if conditions (1) - (4) of section 2 are satisfied, then (3.7) is linear, time invariant, controllable, and has the same spectrum as the equivalent system in the original coordinates.

Theorem 3.9: If (2.1) is CIL, and all nonlinear and time varying entries of f_i are in the image of B_i and if $\int_{-\infty}^{t} a(q)\, dq = G z^1$ for appropriate G then the equivalent system is linear and time invariant.

Theorem 3.10: Equation (3.7) is controllable if the original system (2.1) is CIL. Further, since each component is controllable (3.8) will contain no fixed modes with regard to decentralized state feedback [10].

Theorem 3.11: The spectrum of the equivalent composite system in the original coordinates equals the spectrum of $\hat{A}_{eq} = [A^{11} - A^{12} B^2 (S^1 - G)]$

3.2 PHASE TWO: CONTROL LAW DESIGN.

After selecting the switching surfaces, and hence the equivalent system, the VSC design procedure requires selecting control gains (2.6) which meet the sufficiency conditions for the existence and reachability of a sliding mode.

Due to the presence of the integral term in (2.7) the variable structure control law can be designed at the component level. In this portion of the design the components are effectively decoupled.

We will consider the control law design for two classes of nonlinearities.

Define class I to be the set of all nonlinear terms such that

$$-\infty < (f_i^j)^{min} \le f_i^j(x_i,t) \le (f_i^j)^{max} < \infty \tag{3.12}$$

for all finite values of x_i. Define class II to be the set of all nonlinear terms such that

$$-\infty < (a_i^{j,k})^{min} \le a_i^{j,k}(x_i,t) \le (a_i^{j,k})^{max} < \infty \tag{3.13}$$

for all x_i. Note that if we make the reasonable assumption that off of the switching surfaces the local subsystem (2.1) is Lipshitz, then the conditions for class I systems are automatically satisfied.

CONTROL LAW DESIGN FOR CLASS I NONLINEARITIES. If SB = I and the nonlinearities are of class I, then the control law has the form of (2.6) [1].

Since $S_i B_i = I$ then $\sigma_i^T \dot{\sigma}_i$ is diagonal, hence the design can be treated as m_i scalar designs. For this case a valid sufficient condition for the existence and global reachability of a sliding mode is that

$$\dot{\sigma}_i^j \sigma_i^j < 0 \tag{3.14}$$

for all $x_i \in X_i$ and all t, j=1,...,m_i, i=1,...,N. Note that

$$\dot{\sigma}_i = S_i f_i(x_i,t) + u_i . \tag{3.15}$$

Thus, (3.14) is satisfied whenever

$$u_i^{j,+} < -S_i^j f_i(x_i,t)$$
$$u_i^{j,-} > -S_i^j f_i(x_i,t) \qquad (3.16)$$

where S_i^j is the j−th row of S_i.

Theorem 3.17: Suppose the i−th local subsystem is given by (2.1) where each entry in $f_i(x_i,t)$ is bounded, and the switching surfaces are given by (2.6) and the control law (2.6),(3.16). Then the i−th local state trajectory either intercepts the intersection of the switching surfaces in finite time or approaches it asymptotically.

The control law (3.16) guarantees convergence of the state trajectory to the sliding surface. For systems in which the nonlinear A(x,t) matrix has bounded entries, a linear feedback control law can be found.

CONTROL LAW DESIGN FOR CLASS II NONLINEARITIES. If $S_i B_i = I$ and the nonlinearities are of class II we may use the control law

$$u_i(x_i,t) = K_i x_i(t) , \qquad K_i = [k_i^{j,r}] \qquad (3.18)$$

where

$$k_i^{j,r} = \begin{cases} k_i^{j,r,+} & \text{if } x_i^r \sigma_i^j > 0 \\ k_i^{j,r,-} & \text{if } x_i^r \sigma_i^j < 0 \end{cases}$$

Using the sliding condition (3.14), we note that

$$\sigma_i^T \dot{\sigma}_i = \sigma_i^T [\, S_i A_i(x_i,t) + K_i \,] x_i . \qquad (3.19)$$

Using the properties of the vec operator [9]

$$\mathrm{vec}(\sigma_i^T \dot{\sigma}_i) = x_i^T \otimes \sigma_i^T [\, \mathrm{vec}(S_i A_i(x_i,t) + \mathrm{vec}(K_i) \,] \qquad (3.20)$$

where ⊗ is the Kronecker product of x_i^T and σ_i^T. Thus the sliding condition becomes

$$(\, x_i^T \otimes \sigma_i^T)\mathrm{vec}(K_i) < -(\, x_i^T \otimes \sigma_i^T \,)\mathrm{vec}(S_i A_i(x_i,t)) . \qquad (3.21)$$

Let

$$x_i^T \otimes \sigma_i^T = \left[x_i^1\sigma_i^2 , x_i^1\sigma_i^2 , \cdots , x_i^{n_i}\sigma_i^{m_i}\right] = \left[d_1, d_2, \cdots, d_{nm}\right] \qquad (3.22)$$

$$\mathrm{vec}(K_i) = \left[k_i^{1,1}, k_i^{2,1}, \cdots, k_i^{m_i,1}, \cdots, k_i^{m_i,n_i}\right]^T = \left[\gamma_1, \cdots, \gamma_{n_i m_i}\right]^T$$

$$\mathrm{vec}(S_i A_i) = \left[(S_i A_i)^{1,1}, (SA)_i^{2,1}, \cdots (SA)_i^{m_i,1}, \cdots, (SA)_{i_{m_i,n_i}}\right]^T = \left[\alpha_1, \cdots, \alpha_{n_i m_i}\right]^T .$$

With these equivalences, (3.21) reduces to

$$\left[d_1, \cdots, d_{nm}\right] \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_{n_i m_i} \end{bmatrix} < -\left[d_1, \cdots, d_{n_i m_i}\right] \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{n_i m_i} \end{bmatrix} \qquad (3.23)$$

By requiring each term in the summation of $\sigma_i^T \dot{\sigma}_i < 0$ to be negative for all x_i and t, (3.14) and (3.21) are clearly satisfied, thus

$$d_r \gamma_r < -\, d_r \alpha_r \quad \text{for all } d_r, \alpha_r, \; r = 1, \cdots, n_i m_i . \qquad (3.24)$$

If $d_r > 0$ then $\gamma_r = \gamma_r^+ < -\alpha_r$ for all α_r. If $d_r < 0$ then $\gamma_r = \gamma_r^- > -\alpha_r$ for all α_r. This yields a very simple algorithm; that is set

$$\gamma_r^+ < -\text{Max}(\alpha_r) = \text{Min}(-\alpha_r)$$
$$\gamma_r^- > -\text{Min}(\alpha_r) = \text{Max}(-\alpha_r) \tag{3.25}$$

for all $r=1, \cdots, n_i m_i$, where $\text{Max}(\alpha) = \text{vec}(S_i \text{Max}(A_i))$ where $\text{Max}(A_i) = [a_i^{j,r^{max}}]$ and $\text{Min}(A_i) = [a_i^{j,r^{min}}]$ then set $K_i = \text{vec}^{-1}(\gamma)$.

Theorem 3.26: Suppose the i–th local subsystem (2.1) where each entry in $A_i(x_i,t)$ is bounded, is controlled by the control law (3.18),(3.25) with switching surfaces given by (2.7), then the i–th local state trajectory either intercepts the intersection of the switching surfaces in finite time or approaches it asymptotically.

4. EXAMPLE:

Consider the following system.

Component #1

$$\dot{x}_1 = \begin{bmatrix} x_1^2 \\ x_1^3 \\ -\sin(x_1^2) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (a_1 + u_1)$$
$$b_1 = [\,0\,,\,0\,,\,1\,]\,x_1 \tag{4.1}$$

Component #2

$$\dot{x}_2 = \begin{bmatrix} x_2^1 + 2x_2^2 - x_2^4 \\ \sin(x_2^1) - x_2^2 + x_2^3 \mid x_2^3 \mid \\ \tan^{-1}(x_2^3) + (x_2^4)^2 \\ x_2^2 + 2x_2^3 + x_2^4 \end{bmatrix} x_2 + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} (a_2 + u_2)$$
$$b_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_2 \tag{4.2}$$

With interconnections

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_2^2 \end{bmatrix} \tag{4.3}$$

To determine whether dynamic compensation is needed, investigate

$$\int_{-\infty}^{t} LCx(q)dq = \int_{-\infty}^{t} \begin{bmatrix} x_2^4 \\ 2x_2^1 \\ x_1^2 \end{bmatrix} dq \tag{4.4}$$

Since $x_1^2 = \dot{x}_1^1$ component #1 does not need dynamic compensation. For component #2,

$$\begin{bmatrix} \dot{x}_2^1 \\ \dot{x}_2^4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 & -1 \\ 2 & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.5}$$

Define

$$\dot{x}_2^5 = x_2^3$$
$$\dot{x}_2^6 = x_2^4 \tag{4.6}$$

then (4.4) becomes

$$\int_{-\infty}^{t} \begin{bmatrix} x_2^4 \\ 2x_2^1 \\ x_1^2 \end{bmatrix} dq = \begin{bmatrix} x_2^6 \\ 2x_2^1 - 4x_2^4 + 8x_2^5 + 6x_2^6 \\ x_1^1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 & -4 & 8 & 6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x_2 = G' x_2. \tag{4.7}$$

Thus the compensated system is given by (4.1) and

$$\dot{x}_2 = \begin{bmatrix} x_2^1 + 2x_2^2 - x_2^4 \\ \sin(x_2^1) - x_2^2 + x_2^3 \,|\, x_2^3\,| \\ \tan^{-1}(x_2^3) + (x_2^4)^2 \\ x_2^2 + 2x_2^3 + x_2^4 \\ x_2^3 \\ x_2^4 \end{bmatrix} x_2 + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} (a_2 + u_2)$$

$$b_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} x_2 \qquad (4.9)$$

With interconnections given by (4.3).

The structure of the switching surface is chosen to be of the form

$$\sigma(x) = Sx - \int_{-\infty}^{t} SBLCx(q)\,dq\,. \qquad (4.11)$$

To simplify the design choose S so that SB = I in which case

$$\sigma(x) = Sx - \int_{-\infty}^{t} LCx(q)\,dq = Sx - G'x\,. \qquad (4.12)$$

PHASE 1: CONSTRUCTION OF THE SWITCHING SURFACES

In order to determine a set of acceptable switching surfaces make a series of transformations $z_1 = T_1 x_1$ and $z_2 = T_2 x_2$. Since the first component is already in the proper form then $T_1 = I$, also select $S_1^2 = 1$. Choose

$$T_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (4.13)$$

then

$$B_2^2 = I_2\,, \quad S_2^2 = I_2 \qquad (4.14)$$

and

$$A_2^{11} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \quad A_2^{12} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 0 & 0 \\ 2 & 0 \end{bmatrix}. \qquad (4.15)$$

The equivalent interconnection matrix is $L_{eq} = G$, where $G_1 = G'T^{-1}$ and $G = G_1$ with the appropriate columns removed. Thus

$$L_{eq} = G = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -4 & 8 & 6 & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (4.16)$$

Using the numerical methods of [7,9] the surface matrices, in the original coordinates are:

$$S_1 = \begin{bmatrix} 1.97343\,,\ 0.446225\,,\ 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 2.08781 & 1 & 0 & 1.62092 & -0.64330 & 1.44549 \\ -2.01107 & 0 & 1 & 1.09687 & 0.16347 & 0.38498 \end{bmatrix} \qquad (4.17)$$

which gives a spectrum of $\{-1 \pm j\,,\ -1.2\,,\ -2\,,\ -0.7 \pm 0.5\}$.

CONTROL LAW DESIGN

If we make the assumption that $|x_2^3| < 10$ and $|x_2^4| < 10$, then the algorithm developed in section 3 may be used. Using (3.18),(3.25) a control law which satisfies the sliding conditions (3.14) is

$$u_1 = K_1 x_1$$
$$u_2 = K_2 x_2 \tag{4.18}$$

where

$$k_1^1 = \begin{cases} -1 & \text{if } x_1^1\sigma_1 > 0 \\ 1 & \text{if } x_1^1\sigma_1 < 0 \end{cases} \quad k_1^2 = \begin{cases} -4 & \text{if } x_1^2\sigma_1 > 0 \\ 2 & \text{if } x_1^2\sigma_1 < 0 \end{cases} \quad k_1^3 = \begin{cases} -4 & \text{if } x_1^3\sigma_1 > 0 \\ 2 & \text{if } x_1^3\sigma_1 < 0 \end{cases} \tag{4.19}$$

and

$$k_2^{11} = \begin{cases} -4 & \text{if } x_2^1\sigma_2^1 > 0 \\ 0 & \text{if } x_2^1\sigma_2^1 < 0 \end{cases} \quad k_2^{21} = \begin{cases} 0 & \text{if } \sigma_2^2 x_2^1 > 0 \\ 3 & \text{if } \sigma_2^2 x_2^1 < 0 \end{cases}$$

$$k_2^{12} = \begin{cases} -5 & \text{if } \sigma_2^1 x_2^2 > 0 \\ 0 & \text{if } \sigma_2^1 x_2^2 < 0 \end{cases} \quad k_2^{22} = \begin{cases} 0 & \text{if } \sigma_2^2 x_2^2 > 0 \\ 3.5 & \text{if } \sigma_2^2 x_2^2 < 0 \end{cases} \tag{4.20}$$

$$k_2^{13} = \begin{cases} -13 & \text{if } \sigma_2^1 x_2^3 > 0 \\ 0 & \text{if } \sigma_2^1 x_2^3 < 0 \end{cases} \quad k_2^{23} = \begin{cases} -4.5 & \text{if } \sigma_2^2 x_2^3 > 0 \\ 0 & \text{if } \sigma_2^2 x_2^3 < 0 \end{cases}$$

$$k_2^{14} = \begin{cases} -1.5 & \text{if } \sigma_2^1 x_2^4 > 0 \\ 0 & \text{if } \sigma_2^1 x_2^4 < 0 \end{cases} \quad k_2^{24} = \begin{cases} -14 & \text{if } \sigma_2^2 x_2^4 > 0 \\ 7 & \text{if } \sigma_2^2 x_2^4 < 0 \end{cases}$$

$$k_2^{15} = \begin{cases} -0.5 & \text{if } \sigma_2^1 x_2^5 > 0 \\ 0.5 & \text{if } \sigma_2^1 x_2^5 < 0 \end{cases} \quad k_2^{25} = \begin{cases} -0.5 & \text{if } \sigma_2^2 x_2^5 > 0 \\ 0.5 & \text{if } \sigma_2^2 x_2^5 < 0 \end{cases}$$

$$k_2^{16} = \begin{cases} -0.5 & \text{if } \sigma_2^1 x_2^6 > 0 \\ 0.5 & \text{if } \sigma_2^1 x_2^6 < 0 \end{cases} \quad k_2^{26} = \begin{cases} -0.5 & \text{if } \sigma_2^2 x_2^6 > 0 \\ 0.5 & \text{if } \sigma_2^2 x_2^6 < 0 \end{cases}$$

The results of a computer simulation with initial conditions of $x_1(0) = [1,0,0]^T$, $x_2(0) = [1,0,0,0,0,0]^T$ are given in figures 1 through 3.

5. REFERENCES

[1] V.I. Utkin, *Sliding Modes and Their Application in Variable Structure Systems* , MIR Publishers, Moscow, 1978

[2] V.I. Utkin and K-K.D. Yang, "Methods for Construction of Discontinuity Planes in Multidimensional Variable Structure Systems", Auto. Remote Contr., 1978, pp 1466-1470

[3] P. Hawley and R.DeCarlo, "Variable Structure Control of Interconnected Systems With Applications to Power Systems", Purdue University, School of Electrical Engineering Technical Report TR-EE-83-9

[4] O.M.E. El-Ghezawi, A.S. Zinober, and S.A. Billings "Analysis and Design of Variable Structure Systems Using a Geometric Approach" *Int. J. Control* Vol. 38, No. 3, 1983, pp 657-671

[5] S.Lefebvre, S.Richter, and R.DeCarlo, "Decentralized Variable Structure Control Design for a Two-Pendulum System", IEEE *Trans. Automat. Contr.* , Vol AC-28 No. 12 Dec. 1983, pp.

[6] S.Richter, S.Lefebvre, and R.DeCarlo,"Control of a Class of Nonlinear Systems by Decentralized Control", IEEE *Trans. Automat. Contr*, Vol AC-27 No. 2, April 1982, pp.

[7] S. Lefebvre, S. Richter, and R. DeCalro "A Continuation Algorithm for Eigenvalue Assignment by Decentralized Constant Output Feedback" *Automatica*,

[8] R.A.DeCarlo and R.Saeks, *Interconnected Dynamical Systems* , Marcel Dekker,, New York, 1981

[9] T.L. Harris and R.A. DeCarlo, "Numerical and Theoretical Investigations Into a Continuation Approach To Decentralized Control", Purdue University, School of Electrical Engineering Technical Report TR-EE-82-11

[10] R. Saeks "On the Decentralized Control of Interconnected Dynamical Systems", IEEE *Trans. Automat. Contr*, Vol AC-24 No2, April 1979, pp 269-271

[11] R. G. Morgan and U. Ozguner, "A Decentralized Variable Structure Control Algorithm For Robotic Manipulators", *Proceedings of the 21 Allerton Conference*, Oct. 1983, pp. 315-324

[12] G.H. Goulb and C.F. Van Loan *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 1983

[13] R. Sommer, "Control design for multivariable non-linear time-varying systems", *Int. J. Control*, 1980, Vol. 31, No. 5, pp 883-891

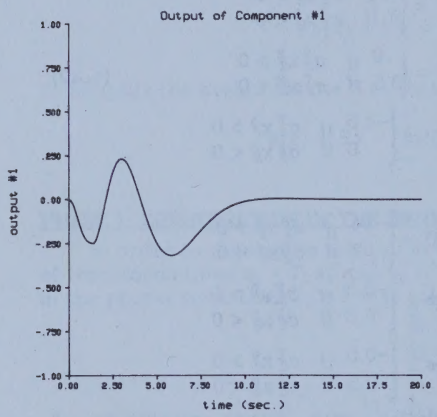


Fig. 1. Output $b_1(t)$ with VSC.

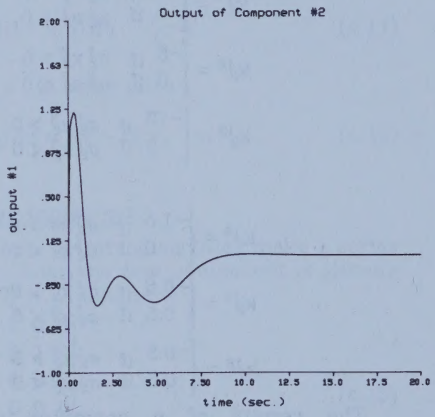


Fig. 2. Output $b_2^1(t)$ with VSC.

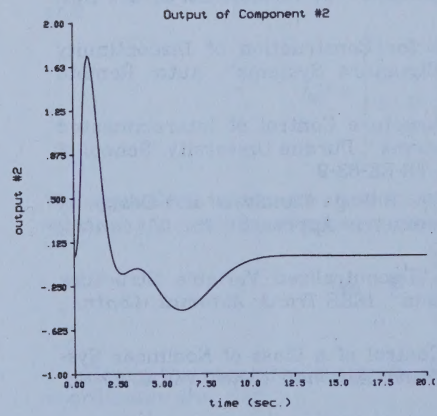


Fig. 3. Output $b_2^2(t)$ with VSC.

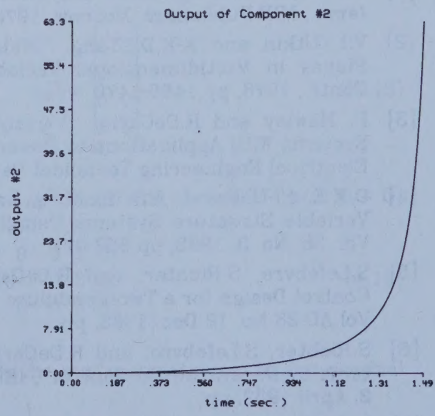


Fig. 4. Output $b_2^2(t)$ without control.

DECENTRALIZED CONTROL USING QUASI BLOCK DIAGONAL DOMINANCE OF TRANSFER FUNCTION MATRICES*

Y. Ohta
Fukui University
Fukui 910, Japan

D. D. Šiljak
University of Santa Clara
Santa Clara, California 95053

T. Matsumoto
Fukui University
Fukui 910, Japan

ABSTRACT

The objective of this paper is to present a generalization of the Nyquist array method to block-wise decompositions, which is based upon a new version of quasi block diagonal dominance conditions. The added flexibility of the proposed method in partitioning the system matrices, allows for a treatment of the system as a collection of weakly coupled subsystems. Then, the controllers of subsystems are designed independently of each other, so that their union represents a decentralized controller for the overall system.

1. INTRODUCTION

The Nyquist and inverse Nyquist array method proposed by Rosenbrock[1], has evolved over past several years as one of the most effective tequniques for the design of controllers of linear multivariable systems. One of the reasons for this development is the fact that the controllers are designed decentrally resulting in a procedure of outstanding conceptual and numerical simplicity. The method is especially efficient in the case of SISO (single-input and single-output) subsystems with weak interactions. If the interactions are not weak, however, an additional effort is required to design pre-controller and a post-controller for overall system so that the overall system has weak interactions. The additional effort may turn out to be quite difficult and time consuming, so that the method may loose its appeal even if the systems of modest size are considered.

To circumvent this difficulty, Bennett and Baras[2] extended Rosenbrock's method using the notion of block diagonal dominance introduced by Feingold and Varga[3]. Subsequently, Limebeer[4] proposed a modification of the method based upon the concept of generalized block diagonal dominance introduced originally by Fiedler[5]. While the stability conditions of [2] allow for independent design of controllers for each subsystem, the conditions are overly conservative. The results of [4] are less conservative than those of [2], but they require a stability test to be performed on the system as a whole, which may be unattractive in view of the decentralized controller design. The earlier results of Araki and Nwokah[6] are more promising than those of [2] and [4], but they are restricted to the case of SISO subsystems.

The purpose of this paper to propose several stability criteria with reduced conservativeness, which are suitable for local design of decentralized controllers. For the large part, the criteria are generalization of those in [6] to the case of MIMO (multi-input-multi-output) subsystems. Moreover, we derive new bounds on the closed-loop transfer function matrix when the k-th block of loops are open. Then, we show how the obtained bounds can be used in a decentralized control system

* The research presented in this paper has been supported in part by the National Science Foundation under the Grant ECS-8315327 and the Grant-in-Aid for the Scientific Research of 1984 (59750313) from Ministry of Education, Science, and Culture of Japan.

design, and thereby provided a solution to what is considered as one of the important outstanding problems in the application of the Nyquist array method to control systems with MIMO subsystems.

Notation. With obvious exceptions, lower case Roman letters denote vectors, and capital Roman letters denote matrices. As is customary, $\mathcal{R}(s)$ stands for the set of rational functions in s with real coefficients. We use $\mathcal{S}$ for the subset of $\mathcal{R}(s)$ consisting of proper rational functions whose poles are all in the open left half-plane. Thus $\mathcal{S}$ is the set of transfer functions of bounded input bounded output stable systems. $\mathrm{Mat}[\mathcal{R}(s)]$ (resp. $\mathrm{Mat}(\mathcal{S})$) denotes the set of matrices with elements in $\mathcal{R}(s)$ (resp. $\mathcal{S}$). $\rho(\cdot)$ denotes the spectral radius of a matrix, $||\cdot||$ is used to denote both a matrix norm and a vector norm. For a positive integer n, $\mathcal{N}(n) = \{1, 2, \ldots, n\}$. A matrix A is denoted by $A = [A_{ij}]$ when A is a block matrix, and the dimension of each block A_{ij} is obvious from the context. For a matrix $A = (a_{ij})$ (resp. a block matrix $A = [A_{ij}]$), $I = \{i1, i2, \ldots, ir\}$, and $J = \{j1, j2, \ldots, jm\}$, A_{IJ} denotes a matrix whose (k,ℓ)-element is $a_{ikj\ell}$ (resp. a block matrix whose (k,ℓ)-block is $A_{ikj\ell}$). If $I = \{k\}$, then we represent A_{IJ} by A_{kJ}. Similarly for an n-vector x and $I = \{i1, i2, \ldots, ir\}$, x_I denotes an r-vector whose k-th component is x_{ik}. I stands for the identity matrix, and 0 represents everything that is zero. For matrices A and B, and vectors x and y, $A \geq B$ and $x \geq y$ means component-wise inequalities. Throughout the paper, an extensive use is made of the properties of M-matrices, which can be found, for example, in [7] and [8]. Due to the limitation in space, all proofs are omitted, but can be obtained upon request from the authors.

2. QUASI BLOCK DIAGONAL DOMINANCE

In this section, we give a brief overview of notion of quasi block diagonal dominance of matrices, which plays a key role in this paper.

Let us consider an mxm complex matrix Q, which is decomposed as

$$Q = Q_D + Q_C \tag{2.1}$$

where

$$Q_D = \mathrm{diag}\{Q_1, Q_2, \ldots, Q_n\}\ , \tag{2.2}$$

$$Q_C = \begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1n} \\ Q_{21} & Q_{22} & \cdots & Q_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ Q_{n1} & Q_{n2} & \cdots & Q_{nn} \end{bmatrix} \tag{2.3}$$

Q_i and Q_{ij} are, respectively, $m_i x m_i$ and $m_i x m_j$ submatrices of Q_D and Q_C, and $m = m_1 + m_2 + \ldots + m_n$.

(2.4)**DEFINITION.** A matrix Q is said to be quasi block diagonal dominant (Q.B.D.D.) if the nxn matrix $W = (w_{ij})$ with elements

$$w_{ij} = \begin{cases} 1 - ||Q_{ii}Q_i^{-1}|| & , i = j \\ - ||Q_{ij}Q_j^{-1}|| & , i \neq j \end{cases} \tag{2.5}$$

is an M-matrix.

From the properties of M-matrices, we have the following :

(2.6)**LEMMA.** Let $B = (b_{ij}) = I - W$, that is, $b_{ij} = ||Q_{ij}Q_j^{-1}||$. The following conditions are equivalent:

(i) There exists a matrix norm $||\cdot||$ such that $||Q_CQ_D^{-1}|| < 1$;

(ii) $\rho(B) = \lambda_{PF}(B) < 1$, where $\lambda_{PF}(B)$ denotes the Perron-Frobenius root of B;

(iii) $W = I - B$ is an M-matrix;

(iv) There exist positive numbers d_j, $j \in \mathcal{N}(n)$, such that

$$d_j^{-1} \sum_{i=1}^{n} d_i b_{ij} < 1 \quad \forall\, j \in \mathcal{N}(n) \tag{2.7}$$

(v) There exist positive numbers d_i, $i \in \mathcal{N}(n)$, such that

$$d_i^{-1} \sum_{j=1}^{n} d_j b_{ij} < 1 \quad \forall\, i \in \mathcal{N}(n). \tag{2.8}$$

Since $Q = (I+Q_CQ_D^{-1})Q_D$, from Lemma (2.6), the following result is automatic:

(2.9)**LEMMA.** If Q is Q.B.D.D., then $\det(Q_D + \mu Q_C) \neq 0$ for all $\mu \in [0,1]$.

Several remarks are in order:

(i) The conditions (2.7) and (2.8) with $d_i = 1$ for all $i \in \mathcal{N}(n)$, define the notion of block diagonal dominance introduced by Robert[9] and Pearce [10]. Okuguchi[11] generalized this notion by introducing the numbers d_i. From the view of Lemma (2.6), Definition (2.4) is equivalent with that of Okuguchi[11]. In [12], a slightly more general definition of Q.B.D.D. than that of (2.4) was introduced.

(ii) Feingold and Varga[3] and Fiedler[5] difined the notion of block diagonal dominance as: A matrix Q is said to be strictly diagonal dominant if there exist positive numbers d_j, $j \in \mathcal{N}(n)$, such that

$$d_j^{-1} \sum_{i\neq j}^{n} d_i ||Q_{ij}|| \; ||Q_j^{-1}|| < 1 \quad \forall\, j \in \mathcal{N}(n) \tag{2.11}$$

Since $||Q_{ij}|| \; ||Q_j^{-1}|| \geq ||Q_{ij}Q_j^{-1}||$, this definition is more conservative than that of Okuguchi[11] as long as we are interested in nonsingularity of $Q_D + \mu Q_C$. We note that in (2.11), it is assumed that $Q_{ii} = 0$ for all $i \in \mathcal{N}(n)$. In [3], $d_i = 1$ for all $i \in \mathcal{N}(n)$.

3. STABILITY

In this section, we consider stability of feedback systems $\mathcal{S}$, described by

Plant $\mathcal{P}$: $y = G(s)e$ (3.1)

Controller $\mathcal{C}$: $w = K(s)z$ (3.2)

Interconnection $\mathcal{J}$: $e = u + w, \quad z = v - y$ (3.3)

where matrices G(s) and K(s) are, respectively, mxr and rxm matrices in $\mathrm{Mat}[\mathcal{R}(s)]$.

We decompose $\mathcal{S}$ as

$$\mathcal{P} : y_i = G_{ii}(s)e_i + \sum_{j\neq i}^{n} G_{ij}(s)e_j, \quad i \in \mathcal{N}(n) \tag{3.4}$$

$$\mathcal{C} : w_i = K_{ii}(s)z_i + \sum_{j\neq i}^{n} K_{ij}(s)z_j, \quad i \in \mathcal{N}(n) \tag{3.5}$$

$$\mathcal{J} : e_i = u_i + w_i, \quad z_i = v_i - y_i, \quad i \in \mathcal{N}(n) \tag{3.6}$$

where $G_{ij}(s)$ and $K_{ij}(s)$ are, respectively submatrices of $G(s)$ and $K(s)$, and their dimensions are, respectively, $m_i x r_j$ and $r_i x m_j$.

The system equation of S is given by

$$S : \begin{bmatrix} I_m & G(s) \\ -K(s) & I_r \end{bmatrix} \begin{bmatrix} z \\ e \end{bmatrix} = \begin{bmatrix} v \\ u \end{bmatrix} . \tag{3.7}$$

Suppose that the feedback systems S is well-posed, that is, $\det[I_m + G(s)K(s)] \not\equiv 0$. Let $H(G,K)(s)$ be the transfer function matrix of S relating (v,u) to (z,e) , that is,

$$H(G,K)(s) = \begin{bmatrix} I_m & G(s) \\ -K(s) & I_r \end{bmatrix}^{-1} . \tag{3.8}$$

We say that S is stable if $H(G,K)(s) \in \mathrm{Mat}(\mathcal{A})$ in which case we say that C stabilizes P .

The following result was proved by Bennett and Baras[2], which is a block-matrix analogue of the Rosenbrock's result[1].

(3.9)**THEOREM.** Let $R(s) = R_D(s) + R_C(s)$, where

$$R_D(s) = \mathrm{diag}\,\{R_1(s), R_2(s), \ldots, R_n(s)\}$$

$$= I_m + \mathrm{diag}\,\{G_{11}(s)K_{11}(s), G_{22}(s)K_{22}(s), \ldots, G_{nn}(s)K_{nn}(s)\} \tag{3.10}$$

$$R_C(s) = [\, R_{ij}(s) \,] = G(s)K(s) - G_D(s)K_D(s) \tag{3.11}$$

$$G_D(s) = \mathrm{diag}\,\{G_{11}(s), G_{22}(s), \ldots, G_{nn}(s)\} \tag{3.12}$$

and

$$K_D(s) = \mathrm{diag}\,\{K_{11}(s), K_{22}(s), \ldots, K_{nn}(s)\} . \tag{3.13}$$

Assume the following:

(i) $G_D(s)K_D(s)$ and $G(s)K(s)$ have the same number of poles in CRH;

(ii) $H(G_D,K_D)(s)$ belongs to $\mathrm{Mat}(\mathcal{A})$;

(iii) $\det[R_D(s) + \mu R_C(s)] \neq 0 \qquad \forall\, s \in D, \forall\, \mu \in [0,1]$, (3.14)

where D is the standard Nyquist D-contour.

Then, S is stable, that is, $H(G,K)(s)$ belongs to $\mathrm{Mat}(\mathcal{A})$.

In Theorem (3.9), CRH denotes the closed right half of the complex plane. The number of poles of $G_D(s)K_D(s)$ in CRH is the sum of poles of $G_D(s)$ and $K_D(s)$ in CRH.

Now, a question arises: Under what conditions, does the condition (3.14) hold ? On the basis of the concept of block diagonal dominance (which was introduced by Feingold and Varga[3] and Fiedler[5]) Bennett and Baras[2], Limebeer[4], Hung and Limebeer[13], and Limebeer and Hung[14] derived sufficient conditions for (3.14). Here, we show improved versions of their results, by using Definition (2.4) of block diagonal dominance.

From Lemmas (2.6) and (2.9), we have:

(3.15)**THEOREM.** Assume the following:

(i) The conditions (i) and (ii) of Theorem (3.9) hold ;

(ii) The Perron-Frobenius root $\lambda_{PF}[B(s)]$ of the matrix $B(s) = [b_{ij}(s)]$ is less than one for all $s \in D$, where

$$b_{ij}(s) = \| R_{ij}(s) R_j^{-1}(s) \| . \tag{3.16}$$

Then, S is stable.

The definition of block diagonal dominance in [3,5], which is used in the previous papers[2,4,13,14], requires, at least, that $\lambda_{PF}[\hat{B}(s)] < 1$, where $\hat{B}(s) = [\hat{b}_{ij}(s)]$ is given by

$$\hat{b}_{ij}(s) = ||R_{ij}(s)|| ||R_j^{-1}(s)|| \ . \tag{3.17}$$

Obviously, $\hat{b}_{ij}(s) \geq b_{ij}(s) \geq 0$ for all i and j, and, hence, our result is less conservative than the previous results .

The condition (ii) of Theorem (3.15) is very useful to analyze stability, but it is not clear how the diagonal blocks $K_{jj}(s)$ affect stability of the overall system S. In the following, we clarify this point under the assumption that $K(s) = K_D(s)$. Then,

$$R_{ij}(s) = G_{ij}(s)K_{jj}(s) \ , \ i \neq j \tag{3.18}$$

and

$$b_{ij}(s) = ||G_{ij}(s)K_{jj}(s)[I_{m_j} + G_{jj}(s)K_{jj}(s)]^{-1}|| \ , \ i \neq j \ . \tag{3.19}$$

Obviously, we have

$$b_{ij}(s) \leq ||G_{ij}(s)|| \ ||K_{jj}(s)[I_{m_j} + G_{jj}(s)K_{jj}(s)]^{-1}|| \ , \tag{3.20}$$

and, if $m_j = r_j$ and $G_{jj}(s)$ is nonsingular, we get

$$b_{ij}(s) \leq ||G_{ij}(s)G_{jj}^{-1}(s)|| \ ||G_{jj}(s)K_{jj}(s)[I_{m_j} + G_{jj}(s)K_{jj}(s)]^{-1}|| \ . \tag{3.21}$$

Let

$$C(s) = [c_{ij}(s)] \ , \quad c_{ij}(s) = \begin{cases} 0 & , \ i = j \\ ||G_{ij}(s)||/||G_{jj}(s)|| & , \ i \neq j \end{cases} \tag{3.22}$$

$$D(s) = \text{diag}\{d_1(s), d_2(s), \ldots, d_n(s)\} \ ,$$

$$d_j(s) = ||G_{jj}(s)|| \ ||K_{jj}(s)[I_{m_j} + G_{jj}(s)K_{jj}(s)]^{-1}|| \ , \tag{3.23}$$

$$\hat{C}(s) = [\hat{c}_{ij}(s)] \ , \quad \hat{c}_{ij}(s) = \begin{cases} 0 & , \ i = j \\ ||G_{ij}(s)G_{jj}(s)|| & , \ i \neq j \ , \end{cases} \tag{3.24}$$

and

$$\hat{D}(s) = \text{diag}\{\hat{d}_1(s), \hat{d}_2(s), \ldots, \hat{d}_n(s)\} \ ,$$

$$\hat{d}_j(s) = ||G_{jj}(s)K_{jj}(s)[I_{m_j} + G_{jj}(s)K_{jj}(s)]^{-1}|| \ . \tag{3.25}$$

Then

$$B(s) \leq C(s)D(s) \tag{3.26}$$

and

$$B(s) \leq \hat{C}(s)\hat{D}(s) \ . \tag{3.27}$$

At this point, we note the following facts:

(i) $\lambda_{PF}[B(s)] < 1$ if and only if I-B(s) is an M-matrix (Lemma (2.6));

(ii) I-B(s) is an M-matrix if either I-C(s)D(s) or $I-\hat{C}(s)\hat{D}(s)$ is an M-matrix[7,8].

Limebeer[4] showed that if $\lambda_{PF}[C(s)D(s)] < 1$, that is, I-C(s)D(s) is an M-matrix, then S is stable. This fact, however, is not useful in a decentralized controller design, because it is not clear how the controllers of individual subsystems affect stability of the overall system S .

To improve on this point, we adopt the idea of [6], which is based on the following facts:

(iii) I-C(s)D(s) (resp. I-$\hat{C}(s)\hat{D}(s)$) is an M-matrix if and only if $D^{-1}(s)$-C(s) (resp. $\hat{D}^{-1}(s)$-$\hat{C}(s)$) is an M-matrix[7,8];

(iv) Let $\Gamma(s)$-C(s) (resp. $\hat{\Gamma}(s)$-$\hat{C}(s)$) be an M_0-matrix. Then, $D^{-1}(s)$-C(s) (resp. $\hat{D}^{-1}(s)$-$\hat{C}(s)$) is an M-matrix if $1/d_j(s) > \gamma_j(s)$ (resp. $1/\hat{d}_j(s) > \hat{\gamma}_j(s)$) for all $j \in \mathcal{N}(n)$[6,7], where

$$\Gamma(s) = \mathrm{diag}\{\gamma_1(s), \gamma_2(s), \ldots, \gamma_n(s)\} , \tag{3.28}$$

and

$$\hat{\Gamma}(s) = \mathrm{diag}\{\hat{\gamma}_1(s), \hat{\gamma}_2(s), \ldots, \hat{\gamma}_n(s)\} . \tag{3.29}$$

From the above facts and Theorem (3.15), we have :

(3.30) **THEOREM.** Assume the following:

(i) The conditions (i) and (ii) of Theorem (3.9) hold ;

(ii) $K(s) = K_D(s)$;

(iii) Let $\Gamma(s)$ be a positive diagonal matrix such that $\Gamma(s)$-C(s) is an M_0-matrix and

$$d_j(s) < 1/\gamma_j(s) , \quad \forall\, j \in \mathcal{N}(n), \quad \forall\, s \in D, \tag{3.31}$$

where C(s), $d_j(s)$ and $\Gamma(s)$ are given by (3.22), (3.23) and (3.28), respectively .

Then, S is stable.

(3.32) **THEOREM.** Assume the following:

(i) The conditions (i) and (ii) of Theorem (3.30) hold ;

(ii) $m_j = r_j$ and $G_{jj}(s)$ is nonsingular for all $j \in \mathcal{N}(n)$ and $s \in D$;

(iii) Let $\hat{\Gamma}(s)$ be a positive diagonal matrix such that $\hat{\Gamma}(s)$-$\hat{C}(s)$ is an M_0-matrix and

$$\hat{d}_j(s) < 1/\hat{\gamma}_j(s) , \quad \forall\, j \in \mathcal{N}(n), \quad \forall\, s \in D, \tag{3.33}$$

where $\hat{C}(s)$, $\hat{d}_j(s)$ and $\hat{\Gamma}(s)$ are given by (3.24), (3.25) and (3.29), respectively.

Then, S is stable.

Several remarks are in order.

(i) The conditions (3.31) and (3.33) are independent of each other.

(ii) If $m_j = r_j = 1$ for all $j \in \mathcal{N}(n)$, then Theorems (3.30), and (3.32) reduce to the result given by Araki and Nwokah[6].

(iii) In general, the condition (iii) of Theorem (3.30) is a sufficient condition for I-C(s)D(s) to be an M-matrix; however, it is also a necessary condition if we choose $\Gamma(s) = \lambda_{PF}[C(s)D(s)]D^{-1}(s)$ as proposed by Limebeer[4]. Therefore, Theorem (3.30) is a generalization of Limebeer's result[4] in the sense that Theorem (3.30) offers added flexibility for the choice of $\Gamma(s)$. Moreover, Theorem (3.30) involves the corresponding result of [2] as a special case : If we choose $\gamma_j(s) = \sum_{i \neq j}^{n} ||G_{ij}(s)||/||G_{ii}(s)||$, then $\Gamma(s)$-C(s) is an M_0-matrix[6,7].

(iv) Suppose that there exist subsets I(i) of $\mathcal{N}(n)$ such that $\mathcal{N}(n) = I(1) \cup I(2) \cup \ldots \cup I(p)$ and that

$$C_{\mathcal{N}(n)\mathcal{N}(n)} = \begin{bmatrix} C_{I(1)I(1)}(s) & 0 & \cdots & 0 \\ C_{I(2)I(1)}(s) & C_{I(2)I(2)}(s) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ C_{I(p)I(1)}(s) & C_{I(p)I(2)}(s) & \cdots & C_{I(p)I(p)}(s) \end{bmatrix} . \tag{3.34}$$

If we choose $\gamma_j(s)$ as $\lambda_{PF}[C_{I(i)I(i)}]$, where i is the index such that $j \in I(i)$, then $\Gamma(s)-C(s)$ is an M_0-matrix[6]; the condition (3.31) of Theorem (3.30) is represented as

$$d_j(s) < 1/\lambda_{PF}[C_{I(i)I(i)}(s)] \ , \ \forall\, j \in I(i), \ \forall\, i \in \mathcal{N}(p). \tag{3.35}$$

Similarly, suppose that there exist subsets I(i) of $\mathcal{N}(n)$ such that $\mathcal{N}(n) = I(1) \cup I(2) \cup \ldots \cup I(p)$ and that

$$\hat{C}_{\mathcal{N}(n)\mathcal{N}(n)} = \begin{bmatrix} \hat{C}_{I(1)I(1)}(s) & 0 & \ldots & 0 \\ \hat{C}_{I(2)I(1)}(s) & \hat{C}_{I(2)I(2)}(s) & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \hat{C}_{I(p)I(1)}(s) & \hat{C}_{I(p)I(2)}(s) & \ldots & \hat{C}_{I(p)I(p)}(s) \end{bmatrix}. \tag{3.36}$$

If we choose $\hat{\gamma}_j(s)$ as $\lambda_{PF}[\hat{C}_{I(i)I(i)}]$, where i is the index such that $j \in I(i)$, then $\hat{f}(s)-\hat{C}(s)$ is an M_0-matrix[6]; the condition (3.33) of Theorem (3.32) is represented as

$$\hat{d}_j(s) < 1/\lambda_{PF}[\hat{C}_{I(i)I(i)}(s)] \ , \ \forall\, j \in I(i), \ \forall\, i \in \mathcal{N}(p). \tag{3.37}$$

(v) Conditions (3.31) and (3.33) are only sufficient conditions for the condition (iii) of Theorem (3.15); however, Theorems (3.30) and (3.32) are very useful in a decentralized controller design, since we can choose $\gamma_j(s)$ and $\hat{\gamma}_j(s)$ independent of K(s) as shown in (iv).

4. DECENTRALIZED CONTROL

In this section, we consider decentralized control problem: Find a suitable set of conditions under which a matrix $K(s) = K_D(s)$ can be chosen to stabilize the system S "piece-by-piece", involving only independent choice of each diagonal block $K_{jj}(s)$ of $K_D(s) = \text{diag}\{K_{11}(s), K_{22}(s), \ldots, K_{nn}(s)\}$. To do this, we first have to generalize certain results by Araki and Nwokah[6] to block-wise decomposition, and derive bounds for k-th diagonal block of a closed-loop transfer function matrix in which the k-th block of loops are open. Then, by using these bounds, we propose a method using the Nyquist array technique[1,6,15] to design controllers of subsystems.

Let us reconsider the feedback system S represented by (3.4)-(3.6). We assume that $K(s) = K_D(s)$. Let $H_k(s)$ be the transfer function matrix relating e_k to y_k where k-th block of loops are open but the other loops are closed. Let $u_i = 0$, $v_i = 0$, for all $j \in \mathcal{N}(n)$,

$$y_j = G_{jj}(s)e_j + \sum_{i \neq j}^{n} G_{ji}(s)e_i, \quad j \in \mathcal{N}(n) \tag{4.1}$$

$$w_j = K_{jj}(s)z_j \ , \ j \in \mathcal{N}(n) : j \neq k \tag{4.2}$$

and

$$e_j = w_j \ , \quad z_j = -y_j \qquad j \in \mathcal{N}(n), \ j \neq k \tag{4.3}$$

Then, $H_k(s) = [h^k_{ij}(s)]$ is defined by

$$h^k_{ij}(s) = y_{ki}/e_{kj} \tag{4.4}$$

where y_{ki} and e_{kj} are, respectively, the i-th component of y_k and the j-th component of e_k.
Now we have

(4.5)**THEOREM.** Given $s \in C$. Assume the following:

(i) $$\det[I_{m_j} + G_{jj}(s)K_{jj}(s)] \neq 0 , \quad \forall\, j \in \mathcal{N}(n) : j \neq k \tag{4.6}$$

(ii) Let $\Gamma(s)$ be a positive diagonal matrix such that $\Gamma(s)-C(s)$ is an M_0-matrix , and

$$d_j(s) < 1/\gamma_j(s) , \quad \forall\, j \in \mathcal{N}(n) : j \neq k \tag{4.7}$$

where $C(s)$, $d_j(s)$, and $\Gamma(s)$ are given by (3.22), (3.23) and (3.28), respectively.

Then,

$$||H_k(s)-G_{kk}(s)|| \leq \gamma_k(s)||G_{kk}(s)|| . \tag{4.8}$$

(4.9)**THEOREM.** Given $s \in C$. Assume the following:

(i) The condition (i) of Theorem (4.5) holds ;

(ii) $m_j = r_j$ and $G_{jj}(s)$ is nonsingular for all j;

(iii) Let $\hat{\Gamma}(s)$ be a positive diagonal matrix such that $\hat{\Gamma}(s)-C(s)$ is an M_0-matrix , and

$$\hat{d}_j(s) < 1/\hat{\gamma}_j(s) , \quad \forall\, j \in \mathcal{N}(n) : j \neq k \tag{4.10}$$

where $\hat{C}(s)$, $\hat{d}_j(s)$, and $\hat{\Gamma}(s)$ are given by (3.24), (3.25) and (3.30), respectively.

Then,

$$||H_k(s)-G_{kk}(s)|| \leq \hat{\gamma}_k(s)||G_{kk}(s)|| . \tag{4.11}$$

If $m_j = r_j = 1$ for all j, then Theorems (4.5), and (4.9) reduce to the result given by Araki and Nwokah[6].

Suppose that there exist subsets I(i) of $\mathcal{N}(n)$, $i \in \mathcal{N}(p)$, such that $\mathcal{N}(n) = I(1) \cup I(2) \cup \ldots \cup I(p)$ and $C(s)$ and $\hat{C}(s)$ are represented by (3.34) and (3.36), respectively. If we choose $\gamma_j(s)$ and $\hat{\gamma}_j(s)$ as $\lambda_{PF}[C_{I(i)I(i)}]$ and $\lambda_{PF}[\hat{C}_{I(i)I(i)}]$, respectively, where i is the index such that $j \in I(i)$, then $\Gamma(s)-C(s)$ and $\hat{\Gamma}(s)-\hat{C}(s)$ are M_0-matrices[6]; the conditions (4.7) and (4.10) are represented, respectively, as

$$d_j(s) < 1/\lambda_{PF}[C_{I(i)I(i)}(s)] , \quad \forall\, j \in I(i) : j \neq k, \quad \forall\, i \in \mathcal{N}(p), \tag{4.12}$$

and

$$\hat{d}_j(s) < 1/\lambda_{PF}[\hat{C}_{I(i)I(i)}(s)] , \quad \forall\, j \in I(i) : j \neq k, \quad \forall\, i \in \mathcal{N}(p). \tag{4.13}$$

Moreover, the bounds (4.8) and (4.11) are given, respectively, by

$$||H_k(s)-G_{kk}(s)|| \leq \lambda_{PF}[C_{I(m)I(m)}(s)]||G_{kk}(s)|| , \tag{4.14}$$

and

$$||H_k(s)-G_{kk}(s)|| \leq \lambda_{PF}[\hat{C}_{I(m)I(m)}(s)]||G_{kk}(s)|| . \tag{4.15}$$

Therefore, our bounds are independent of the choice of controllers.

Bennett and Baras[2] derived a bound of $||\{H_k^{-1}(s)-[I+G_{kk}(s)]\}^{-1}||^{-1}$; Limebeer[4] gave a bound of $||[H_k(s)-G_{kk}(s)]^{-1}||^{-1}$. These results are generalizations of the previous result using Ostrowski band[1], and, hence, these are affected by the choice of controllers of other subsystems.

Now, we consider an application of Theorem (4.9) to the Nyquist array method[1,6,15]. Let

$$K_{jj}(s) = \hat{K}_j(s)F_j(s) , \quad j \in \mathcal{N}(n), \tag{4.16}$$

$$F_j(s) = \mathrm{diag}\{f_1^j(s), \ f_2^j(s), \ldots, \ f_{r_j}^j(s)\}, \quad j \in \mathcal{N}(n), \tag{4.17}$$

$$e_j = w_j = \hat{K}_j(s)x_j, \quad j \in \mathcal{N}(n), \tag{4.18}$$

and

$$x_{ji} = -f_i^j(s)y_{ji} \ , \ j \in \mathcal{N}(n), \ i \in \mathcal{N}(r_j) \ : \ ji \neq k\ell \ , \tag{4.19}$$

where x_{ji} and y_{ji} are the i-th elements of x_j and y_j, respectively.

Moreover, let $h_\ell^k(s)$ be the transfer function relating $x_{k\ell}$ to $y_{k\ell}$ where the ℓ-th loop of the k-th block is open but other loops are all closed. Then we have:

(4.20)**THEOREM.** Given $s \in C$. Suppose that there exist subsets I(i) of $\mathcal{N}(n)$, $i \in \mathcal{N}(p)$, such that $\mathcal{N}(n) = I(1) \cup I(2) \cup \ldots \cup I(p)$ and $\hat{C}(s)$ is represented by (3.36). Assume the following:

(i) the conditions (i) - (iii) of Theorem (4.9) hold, in which $||\cdot||$ is given by $||\cdot||_1$ or $||\cdot||_\infty$;

(ii) $\hat{K}_i(s)$ is nonsingular for all i

(iii) $$b_j^k(s) < [(1+\lambda_{PF}[\hat{C}_{I(m)I(m)}])||T^{kk}(s)||-1]^{-1} \ , \ j \neq \ell \tag{4.21}$$

where m is the index such that $k \in I(m)$,

$$b_j^k(s) = |q_{jj}^{kk}(s)f_j^k(s)(1+q_{jj}^{kk}(s)f_j^k(s))^{-1}| \ , \tag{4.22}$$

$q_{jj}^{kk}(s)$ is the (j,j)-element of $Q_{kk}(s)$,and $Q_{kk}(s)$,and $T^{kk}(s) = [t_{ij}^{kk}(s)]$ are defined, respectively, by

$$Q_{kk}(s)=G_{kk}(s)\hat{K}_k(s) \tag{4.23}$$

$$t_{ij}^{kk}(s) = |q_{ij}^{kk}(s)|/|q_{jj}^{kk}(s)| \ . \tag{4.24}$$

Then,

$$|h_\ell^k(s)-q_{\ell\ell}^{kk}(s)| \leq [(1+\lambda_{PF}[\hat{C}_{I(m)I(m)}])||T^{kk}(s)||-1]|q_{\ell\ell}^{kk}(s)| \ . \tag{4.25}$$

Finally, we show the following result in connection with the condition (4.13).

(4.26)**THEOREM.** Given $s \in C$ and $k \in \mathcal{N}(n)$. Suppose that there exist subsets I(i) of $\mathcal{N}(n)$, $i \in \mathcal{N}(p)$, such that $\mathcal{N}(n) = I(1) \cup I(2) \cup \ldots \cup I(p)$ and $\hat{C}(s)$ is represented by (3.36). Assume the following:

(i) $$1+q_{jj}^{kk}(s)f_j^k(s) \neq 0 \tag{4.27}$$

for all j, where $q_{jj}^{kk}(s)$ is the (j,j)-element of $Q_{kk}(s)$ which is defined by (4.23);

(ii) $\hat{K}_i(s)$ is nonsingular for all i ;

(iii) $$b_j^k(s) < [(1+\lambda_{PF}[\hat{C}_{I(m)I(m)}])||T^{kk}(s)||-1]^{-1} \tag{4.28}$$

for all j, in which $||\cdot||$ is given by $||\cdot||_1$ or $||\cdot||_\infty$, m is the index such that $k \in I(m)$, $b_j^k(s)$ and $T^{kk}(s)$ are given by (4.22) and (4.24), respectively.

Then, we have (4.13).

In connection with Theorem (4.20) and (4.26), we note that none of $\hat{c}_{ij}(s)$ is affected by the choice of $\hat{K}_j(s)$; we can choose $\hat{K}_j(s)$ in any way we want without disturbing interaction index $\lambda_{PF}[\hat{C}(s)]$ as long as $\hat{K}_j(s)$ is nonsingular. This is the fact that opens up real possibility for using Nyquist array method in the design of decentralized control systems.

5. CONCLUSION

In this paper, we presented a generalization of the Nyquist array method to block-wise decompositions of linear systems. The basis for the generalization is the new formulation of the quasi block diagonal dominance. The significance of the generalization is due to the fact that under mild conditions, the choices of the individual subsystem controllers do not affect each other, thus allowing for a decentralized control scheme. In the same spirit, we derive new bounds on the closed-loop transfer functions, which are also invariant to the choice of the subsystems controllers. Using these bounds, we proposed a method relying on the Nyquist array method[1,6, 15] to the decentralized control of MIMO subsystems. Moreover, we should stress that we can choose pre-controllers in any number of ways without disturbing the interaction index, which is an essential factor in decentralized design of MIMO feedback control systems.

REFERENCES

[1] H. H. Rosenbrock, "Computer-Aided Control System Design," Academic Press, New York, 1974.

[2] W. H. Bennett and J. S. Baras, "Block diagonal dominance and design of decentralized compensators," Proc. IFAC Symp. Large Scale System Theory and Application, Toulouse, pp.93-102, 1980.

[3] D. G. Feingold, and R. C. Varga , "Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem," Pacific Journal of Mathematics, vol.12, pp.1241-1250,1962.

[4] D. J. N. Limebeer, "The application of generalized diagonal dominance to linear system stability theory," Int. J. Control, vol.36, pp.185-212, 1982.

[5] M. Fiedler, "Some estimate of spectra of matrices," Symp. on Numerical Treatment of Ordinary Differential, Integral, and Integro-Differental Equations, Birkhauser Verlag, Basel, pp.33-36, 1961.

[6] M. Araki and O. I. Nwokah, "Bounds for closed-loop transfer functions of multivariable systems," IEEE Trans. Auto. Contr., vol.AC-20, pp.666-670, 1975.

[7] M. Fiedler and V. Ptak, "On matrices with non-positive off-diagonal elements and positive principal minors," Czeh. Math. J., vol.12, pp.382-400, 1962.

[8] D. D. Šiljak, Large-Scale Dynamic Systems, North-Holland, New York, 1978.

[9] F. Robert, "Blocs-H-matrices et convergence des methodes iteratives classiques par blocks," Linear Algebra and Its Applications, vol.2, pp.223-265, 1966.

[10] I. F. Pearce, "Matrices with dominating diagonal blocks," Journal of Economic Theory, vol.9, 159-170, 1974.

[11] K. Okuguchi, "Matrices with dominant diagonal blocks and economic theory," Journal of Mathematical Economics, vol.5, pp.43-52, 1978.

[12] Y. Ohta and D. D. Šiljak, "A generalization of block diagonal dominance including overlapping blocks," Proc. of IFAC 9th World Congress, Budapest, Hungary, July 2-6, 1984.

[13] Y. S. Hung and D. J. N. Limebeer, "Robust stability of additively perturbed interconnected system", Cambridge University Engineering Dept. Report. No.CUED/F-CAM/TR-220, 1982.

[14] D. J. N. Limebeer and Y. S. Hung, "Robust stability of interconnected systems," IEEE Trans. Circuits and Systems, vol.CAS-30, pp.397-403, 1983.

[15] M. Araki, K. Yamamoto, and B. Kondo, "GG-pseudo-band method for the design of multivariable control systems," IFAC 8th World Congress, Vol. 3 pp.137-142, Kyoto, Japan, Aug., 1981.

SIMULTANEOUS ROUTING AND SIGNALIZATION IN TRAFFIC NETWORKS

Farshad Khorrami
Department of Math
The Ohio State University
Columbus, Ohio 43201

Umit Ozguner
Dept. of Elec. Eng.
The Ohio State University
Columbus, OH 43201

ABSTRACT

Optimization of traffic (according to some measure) through a congested network encompasses a vital component of many fields. It is especially important in the transportation networks which are discussed throughout this paper.

Although routing and signalization (particularly signalization) have received much attention, the interconnection of these two important elements in traffic networks has not been considered. It is the intent of this paper to develop a mathematical model including the effects of both routing and signalization at the same time in the network. Furthermore, a robust stabilizing feedback law for optimum performance of the network according to a cost function depending upon the delays in the network is to be derived.

1. INTRODUCTION

The major factors in improving and controlling the performance of traffic networks are signalization and routing.

Basically, signalization deals with finding the "best" signal settings for a network. Optimum signal settings are comprised of three factors:

1. Signal split time (red and green time).

2. The duration of the cycle time (i.e., red time and green time).

3. The offset time (i.e., the time difference between setting among adjacent intersections).

The effect of these three factors have been studied in detail by many authors in the past three decades. Studies by Webster [1,2] showed that for fixed time signals (i.e., fixed cycle time), the optimal signalization plan can be found by having the following information about the intersection:

a) Total input flow on each arm.

b) Saturation flows on each arm.

c) Lost time between the red and green time (i.e., clearance time).

Also, it was suggested that vehicle actuated signals perform better than fixed time signals under light and moderate traffic conditions. But as the traffic flow approaches its saturation level, the vehicle actuated signals tend to behave as if they were fixed time signals. However, it was noted that although the fixed time signals set to their optimum value and vehicle actuated signals tend to operate about the same, the vehicle actuated signals have the advantage of adapting to new flow patterns. Furthermore, not being able to have an accurate measurement of the input flow to each arm of an intersection causes degredation in optimum setting of a fixed time signal which in turn may increase the delay in the network substantially.

As Gazis noted [3] the level of improvement that can be made in traffic networks using signal control only has probably been reached. So, one has to resort to other means of controlling the traffic, and that is routing. Further research was in agreement with Gazis remark.

Route guidance is the way to reassign the distribution of the traffic in a network in an optimum manner (by some measure). Basically, this optimization is carried out to obtain either a system-optimizing solution or a user optimizing solution. The first one is to minimize the total delay in the network as a whole, and the latter one is to assign a route to an individual user in order to reach his destination as fast as possible. In this paper, a system-optimizing solution is of interest since this criterion gives the least total delay and an even distribution of traffic in the network (see Akcelik [5] and Dafermos [6]). Recently, P.E. Sarachik and U. Ozguner [7] investigated "decentralized dynamic routing for congested networks". The assumptions made on the network were: single destination, deterministic inputs, and queues are the only elements causing delays in the network. A decentralized routing strategy as a feedback law is given in the paper. The main result pointed out is that the outlined algorithm would be loop free (i.e., no traffic around "cyclic paths"). Also, a formula to estimate the "time to clear" in the network is provided by the authors.

A comprehensive review of route control has been given by Akcelik [8]. In the studies mentioned, it is experimentally (through simulations) observed that if the route and the signal control are applied simultaneously, a reduction in delays and an increase in the throughput of the system will result.

Routing has received less attention than signalization. But, in the past few years, research has been initiated to incorporate route control in the current area traffic control policies. There are programs underway in Japan and West Germany implementing these ideas.

2. PROBLEM IDENTIFICATION AND MODEL DESCRIPTION

It is known that the optimal signal time is dependent upon the link travel times and delays at intersections. But, the link travel times and delays in a network themselves are dependent upon the signal times. Due to a flow perturbation into a system the new optimal signal setting for the perturbed flows have to be computed. Modifying the signal time plan changes the delays in the network and ultimately creates a new distribution and demand of traffic in the system. This new flow pattern, in turn, requires recomputation of the signal time and this process will continue recursively.

It is of interest to know whether this process converges to a steady state solution or ultimately causes deterioration of the network performance.

In order to analyze the interconnection of routing and signalization, a mathematical model relating the two factors is needed. The three major factors in signalization has been dealt with extensively in literature. Of the three controls on signalization, the signal split time is observed to be the most important and effective one in controlling the behavior of the network. So in this paper, only the signal split time was considered.

Let i^{th} intersection be shown as in Fig. 1.

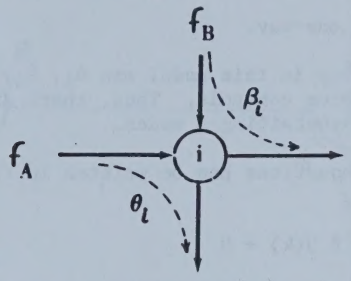


Fig. 1

Let

θ_i = the number of cars turning right at i^{th} intersection between k^{th} and $(K+1)^{th}$ time interval.

β_i = the number of cars turning left at i^{th} intersection between K^{th} and $(K+1)^{th}$ time interval.

∂_x^i = the green time in the x-direction (i.e., direction #1) at K^{th} time interval.

c^i = the cycle time.

ℓ^i = the clearance time for the i^{th} intersection.

f_x^i = The flow rate out of the queues in x direction between K^{th} and $(K+1)^{th}$ time interval.

f_y^i = the flow rate out of the queues in y direction between K^{th} and $(K+1)^{th}$ time interval.

f_A, f_B = External inputs (incoming source A and B).

One can write the system model as:

$$q_x^i(K+1) = q_x^i(K) + f_A - \theta_i - f_x^i \partial_x^i$$
$$q_y^i(K+1) = q_y^i(K) + f_B - \beta_i - f_y^i (c^i - \partial_x^i - \ell^i) \qquad (1)$$

Assumptions made on the network are as follows:

a) All the links are connected through the nodes (i.e., no branches or merges are allowed from links). Merges and branches can be dealth with by adding extra nodes.

b) The cycle time is fixed throughout the network.

c) The flow rates out of queues are dependent upon the characteristic of each node and are assumed to be constants in order to have a linear model.

d) No turn on red.

e) All traffic is one-way.

The control variables in this model are θ_i, β_i, and ∂_x^i's. Each node has two states and three controls. Thus, there are 2n states and 3n controls for a network containing n nodes.

The overall state equations can be written in the following manner:

$$\underline{q}(k+1) = I\underline{q}(k) + B\,U(k) + D \qquad (2)$$

The objective of this paper is to find a robust decentralized feedback law such that the queue lengths are balanced or sent to a desired set point, i.e., clearing the queues by choosing the turn ratios and signal times at each intersection.

Finally, it is desired to find a robust stabilizing feedback law minimizing the total delay in the network according to a cost function (system-optimizing solution).

To stabilize the system governed by (2), it is desired to apply the following control (i.e., state error feedback):

$$U(k) = -K[q(k) - \bar{q} - (BK)^{-1}D] \qquad (3)$$

where $\bar{q}$ is the desired set point, i.e., $q(k) \to \bar{q}$, as $k \to \infty$.

In order to have a decentralized feedback law, K needs to be block diagonal (blocks of 3x2). The following theorem establishes existence of such a block diagonal matrix.

Theorem 1

Let B be a 2n x 3n matrix and P(B) = 2n. Then, there exists a block diagonal matrix K (blocks of 3x2) such that (I+BK) is a stable matrix, i.e., all its eigenvalues are inside the unit disc. Permutation of columns of B is also permitted.

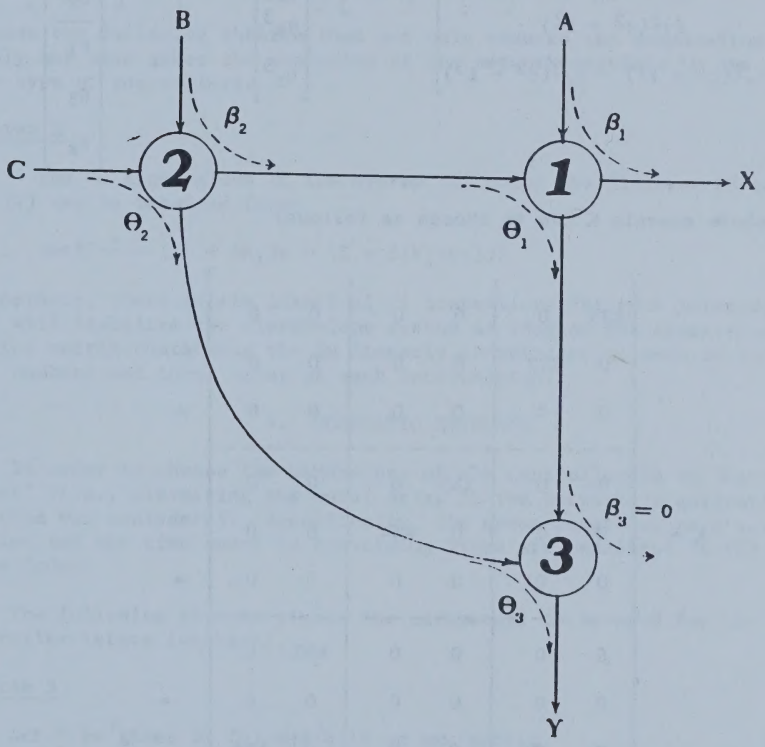
The following Lemma by Corfmat and Morse [9] is used to prove theorem 1.

Lemma 1

If A is an nxn strongly nonsingular matrix, then there exists a diagonal matrix D such that (I + DA) is stable.

Example

Consider the following network:



In this case:

$$B = \left[\begin{array}{ccc|ccc|ccc} -1 & 0 & -f_x^1 & 0 & 1 & f_x^2 & 0 & 0 & 0 \\ 0 & -1 & f_y^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -1 & 0 & -f_x^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & f_y^2 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & -f_y^2 & -1 & 0 & 0 \\ 1 & 0 & -f_y^1 & 0 & 0 & 0 & 0 & 0 & f_y^3 \end{array}\right]$$

$$D = \begin{bmatrix} 0 \\ f_A - f_y^1(c^1 - \ell^1) \\ f_c \\ f_B - f_y^2(c^2 - \ell^2) \\ fy^2(c^2 - f^2) \\ f_y^1(c^1 - \ell^1) - f_y^3(c^3 - \ell^3) \end{bmatrix} \quad ; \quad \underline{q} = \begin{bmatrix} q_x^1 \\ q_y^1 \\ \hline qx2 \\ q_y^2 \\ \hline q_x^3 \\ q_y^3 \end{bmatrix} \quad ; \quad U = \begin{bmatrix} \theta_1 \\ \beta_1 \\ \partial_x^1 \\ \hline \theta_2 \\ \beta_2 \\ \partial_x^2 \\ \hline \theta_3 \\ \beta_3 \\ \partial_x^3 \end{bmatrix}$$

In the above example K can be chosen as follows:

$$K = \left[\begin{array}{cc|cc|cc} K11 & 0 & 0 & 0 & 0 & 0 \\ 0 & K22 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & K33 & 0 & 0 & 0 \\ 0 & 0 & 0 & K44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & K55 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & K66 \end{array}\right]$$

3. PROPORTIONAL-INTEGRAL FEEDBACK

It is well known that proportional-integral feedback may be used to regulate systems under constant unmeasurable disturbances and constant setpoints. The advantage over the approach presented earlier (even if disturbances are measurable) is robustness against parameter variations in the system. The use of this controller is beneficial for the system under consideration since the flow rates and also the input traffic from the external sources are assumed to be constant.

The PI controller will have the structure

$$u(k) = K_1 q(k) + K_2 H(k)$$

$$H(k+1) = H(k) + q(k) - \bar{q} \tag{4}$$

The closed loop system is then

$$\begin{bmatrix} q(k+1) \\ H(k+1) \end{bmatrix} = \begin{bmatrix} I + BK_1 & BK_2 \\ I & I \end{bmatrix} \begin{bmatrix} q(k) \\ H(k) \end{bmatrix} + \begin{bmatrix} D \\ -\bar{q} \end{bmatrix} \tag{5}$$

We have the following theorem that not only reduces the computation considerably but also makes the expansion of the network possible by use of the same type of controllers.

Theorem 2

The 4n eigenvalues of the system (2) under the discrete PS controller (4) can be obtained from

$$\det\{Iz^2 - (2I + BK_1)z + (I + B[K_1 - K_2])\}$$

Furthermore, there exists identical PI controllers for each intersection that will stabilize the closed-loop system as long as the eigenvalues of B* (the matrix containing the 2n linearly independent columns of B) are real numbers and turns occur at each intersection.

4. QUADRATIC REGULATOR

In order to choose the parameters of the controller in an "optimum manner" (i.e., minimizing the total delay in the network) a quadratic cost function was conisdered. Specifically, the queue length at each intersection and the time spent in traversing links are penalized in the performance index.

The following theorem yields the parameters to be used for the controller (state feedback).

Theorem 3

Let Σ be given by (6) and B is an mxn matrix

$$q(k+1) = Iq(k) + Bu(k) \quad \text{and} \quad y(k) = Cq(k) \tag{6}$$

The following control

$$u^*(k) = -B'(BB')^{-1}K^{-1}C'Cq(k)$$

where

$$K = \begin{cases} 2(BB')^{-1}\{[4(BB'C'C)^{-1} + I]^{1/2} - I\}^{-1} & , \text{ if } m \leq n \\ 2B(B'B)^{-1}\{[4(B'C'CB)^{-1} + I]^{1/2} - I\}^{-1}(B'B)^{-1}B' & , \text{ if } m > n \end{cases} \tag{6.a}$$

minimizes the performance index

$$J = 1/2 \sum_{k=0}^{\infty} \left(\|q(k)\|_Q^2 + \|u(k)\|_R^2 \right)$$

where $Q = Q' \geq 0$ and $R = R' > 0$. In particular, $Q = C'C$ and $R = I$.

Remark: Note that the recurrence relation

$$X_{n+1} = 1/2 \, (X_n + DX_n^{-1})$$

Converges to the square root of the positive definite matrix D and may be used to calculate the feedback gains for the controllers.

As mentioned before, a proportional integral feedback is employed for robustness. The closed loop state equations are

$$Z(k+1) = \Phi A(k) + \Gamma w(k) \tag{7}$$

where

$$Z(k) = \begin{bmatrix} q(k+1) - q(k) \\ q(k) - \bar{q} \end{bmatrix}, \quad w(k) = u(k+1) - u(k) \tag{7.a}$$

$$\Phi = \begin{bmatrix} A & 0 \\ C & I \end{bmatrix}, \quad \Gamma = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

In this case A = I.

Theorem 4

Let Σ be given by (7), then the control

$$w^*(k) = P \cdot Z(K) \tag{8}$$

or

$$u^*(k) = P_1 q(k) + P_2 \sum_{n=0}^{k-1} [q(n) - \bar{q}] \tag{8.a}$$

where

$$P = [P_1 \; P_2] = - [I + \Gamma' K \Gamma]^{-1} \Gamma' K \Phi, \quad K = \begin{bmatrix} K_1 & K_2 \\ K_2' & K_3 \end{bmatrix}$$

and K_1, K_2, K_3 satisfy the set of equations

$$\begin{cases} (K_1 + C'K_2')(I + BR^{-1}B'K_1)^{-1}(I - BR^{-1}B'K_2C) + K_2C + C'K_3C = K_1 - Q_1 \\ K_2'(I + BR^{-1}B'K_1)^{-1}BR^{-1}B'K_2 = Q_2 \\ (K_1 + C'K_2')(I + BR^{-1}B'K_1)^{-1}BR^{-1}B'K_2 - C'K_3 = 0 \end{cases} \tag{9}$$

minimizes the performance index, J,

$$J = 1/2 \sum_{k=0}^{\infty} (\|z\|_Q^2 + \|w\|_R^2)$$

where $Q = Q' \geq 0$ and $R = R' > 0$. In particular, $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}$

Furthermore, for the particular weighting $Q_2 = 0$ the optimal control is given as in Eq. (6.a).

5. CONCLUSION

In this paper, effects of applying routing and signalization in a traffic network was considered. A mathematical model interlinking these two elements was introduced and existence of decentralized controllers (i.e., state feedback and proportional integral feedback) to stabilize the system was shown. The discrete-time quadratic regulator was considered and it was shown that the optimal feedback gains can be derived directly.

It should be mentioned that the constraints on the controls, states (queue length), and the capacity of the links were not included at this point. Further research needs to be done to include the effects of constraints and stochastic inputs to the system.

Because of lack of space proofs of theorems are not included and may be obtained from the authors.

6. ACKNOWLEDGMENT

This material is based upon work supported by the Department of Transportation under Contract DTRS 5683-C-00045.

7. REFERENCES

[1] Webster, F.V., Traffic Signals. Conference on Traffic Engineering, Trondheim. Jan., 1961.

[2] Webster, F.V. and Wardrop, J.E., Capacity of Urban Intersections. 6th Internat. Study Week, Traff. Enging. Salzburg, 1962.

[3] Gazis, D.C. "Optimum Control of a System of Oversaturated Intersection," Operations Research, Vol. 12, No. 6, Nov-Dec, 1964.

[4] Davison, E.J. and U. Ozguner, "Decentralized Control of Traffic Net works," IEEE Trans. Auto. Contr., Vol. AC-28, No. 6, June, 1983.

[5] Akcelik, R. and M.J. Maher, "Route Control-Simulation Experiments," Transp. Res., Vol. 11, pp. 25-31, 1977.

[6] Dafermos, S., "An Extended Traffic Assignment Model with Applications to Two-way Traffic," pp. 367-319.

[7] Sarachik, P.E. and U. Ozguner, "On Decentralized Dynamic Routing for Congested Traffic Networks," IEEE Trans. Auto Contr., Vol. AC-27, No. 6, Dec., 1982.

[8] Akcelik, R., "Route Control of Traffic in Urban Road Networks," Transp. Res., Vol. 11, pp. 25-31, 1977.

[9] J.P. Corfmat and A.S. Morse, "Stabilization with Decentralized Feedback Control," IEEE Trans. Auto. Control, pp. 679-682, Dec. 1973.

ANALYSIS OF REDUCED ORDER TWO-TIME-SCALE MODELS

S. J. FU
Kansas Gas and Electric Co.
P. O. Box 208
Wichita, Kansas 67201

M. E. SAWAN AND M. T. TRAN
Electrical Engineering Department
Wichita State University
Wichita, Kansas 67208

ABSTRACT

The structures of singularly perturbed continuous-time and discrete-time systems are compared. Matrix identities and transfer functions derived from equivalent systems are presented in this paper. They can be used as a validity check for discrete-time two-time-scale system models.

DECOMPOSITION OF TWO-TIME-SCALE SYSTEMS

For a linear continuous-time system with two-time-scale property [1]

$$\dot{x}_1 = A_{11}x_1 + A_{12}x_2 + B_1u \tag{1a}$$

$$\mu\dot{x}_2 = A_{21}x_1 + A_{22}x_2 + B_2u \tag{1b}$$

where μ is a small positive scalar. Assuming A_{22} is nonsingular, the slow and fast subsystems and quasi-steady state of the fast state become

$$\dot{x}_S = A_Ox_S + B_Ou_S, \qquad \mu\dot{x}_f = A_{22}x_f + B_2u_f \tag{2a}$$

$$\bar{x}_2 = -A_{22}^{-1}(A_{21}x_S + B_2u_S) \tag{2b}$$

where x_S is the slow state vector, x_f is the fast state vector, $\bar{x}_2$ is the quasi-steady state of the fast state vector, u_S and u_f are slow and fast control with $u = u_S + u_f$, and

$$A_O = A_{11}-A_{12}A_{22}^{-1}A_{21}, \qquad B_O = B_1-A_{12}A_{22}^{-1}B_2 \tag{2c}$$

For discrete-time systems there are several approaches to obtain the singularly perturbed models [2,3]. The model in [2] is used

$$x_1(n+1) = F_{11}x_1(n) + F_{12}x_2(n) + G_1u(n) \tag{3a}$$

$$x_2(n+1) = F_{21}x_1(n) + F_{22}x_2(n) + G_2u(n) \tag{3b}$$

Assuming that $I-F_{22}$ is nonsingular, the slow subsystem, fast subsystem and quasi-steady state of the fast state vector are decomposed as [2]

$$x_S(n+1) = A_Sx_S(n) + B_Su_S(n), \qquad x_f(n+1) = F_{22}x_f(n) + G_2u_f(n) \tag{4a}$$

$$\bar{x}_2(n) = (I - F_{22})^{-1}[F_{21}x_S(n) + G_2u_S(n)] \tag{4b}$$

where x_S is the slow state vector, x_f is the fast state vector, $\bar{x}_2$ is the quasi-steady state of the fast state vector, u_S and u_f are slow and fast control with $u = u_S + u_f$, and

$$A_S = F_{11} + F_{12}(I - F_{22})^{-1}F_{21}, \qquad B_S = G_1 + F_{12}(I - F_{22})^{-1}G_2 \tag{4c}$$

MATRIX IDENTITIES FOR EQUIVALENT SYSTEMS

When a continuous-time system is sampled at a proper rate, an equivalent discrete-time system is obtained. Sampling rate is an important factor to preserve the physical structure and properties of the systems. Hence given equivalent systems (1) and (3), and sampling interval T, it is natural to expect that the two-time-scale property is preserved and that the slow subsystems will be equivalent as well as the fast subsystems. Some matrix identities are established form (2) and (4):

$$A_S = e^{A_OT}, \qquad B_S = \left(\int_0^T e^{A_Ot}dt \right) B_O \tag{5}$$

$$F_{22} = e^{\frac{A_{22}T}{\mu}} , \qquad G_2 = \left(\int_0^T e^{\frac{A_{22}T}{\mu}} dt \right) B_2 \tag{6}$$

For the quasi-steady state of the fast state vectors, the algebraic relationship in (2b) and (4b) is preserved, i.e.

$$(I - F_{22})^{-1} F_{21} = -A_{22}^{-1} A_{21}, \qquad (I - F_{22})^{-1} G_2 = -A_{22}^{-1} B_2 . \tag{7}$$

Identities (5)-(7) reveal some important insight of the two-time-scale system structure and can be used to check the validity of models. However, these identities are obtained by approximations [1,2].

FREQUENCY DOMAIN ANALYSIS

When the singular perturbational approximations for linear systems are developed in the frequency domain, the reduced order system transfer functions are obtained in [4]. The output equation for systems (1) and (3) is given as

$$y = Cx = C_1 x_1 + C_2 x_2 \tag{8}$$

where C_1 and C_2 are n_1-dimension and n_2-dimensional vectors.

For the continuous-time system the transfer function of the reduced order model is

$$H_r(s) = C_o (sI - A_o)^{-1} B_o \tag{9}$$

where $C_o = C_1 - C_2 A_{22}^{-1} A_{21}$

For the discrete-time system the transfer function of the reduced order model is

$$H_r(z) = C_S (zI - A)^{-1} B_S \tag{10}$$

where $C_S = C_1 + C_2 (I - F_{22})^{-1} F_{21}$

It should be noted that (10) is true for different two-time-scale models in [2] and [3]. For equivalent systems, $H_r(z)$ and $H_r(s)$ are related by

$$H_r(z) = \Sigma [\text{residues of } \frac{H_r(s)}{1 - e^{sT} z^{-1}} \text{ at poles of } H(s)] \tag{11}$$

CONCLUSION

Assuming that the system structure and properties are invariant, the two-time-scale property of continuous-time and discrete-time systems is investigated. A set of matrix identities are presented. In general there are some inaccuracies in the identities for the fast subsystems but the identities are very accurate for the slow subsystems which are normally the dominant parts of two-time-scale design techniques.

REFERENCES

[1] J. H. Chow and P. V. Kokotovic, "A Decomposition of Near-Optimal Regulators for Systems with Slow and Fast Modes," IEEE Trans. Automat. Contr., Vol. AC-21, pp. 701-705, 1976.

[2] M. T. Tran and M. E. Sawan, "Reduced Order Discrete-Time Models," Int. J. System Sci., Vol. 14, No. 7, 1983.

[3] G. Blankenship, "Singularly Perturbed Difference Equations in Optimal Control Problems," IEEE Trans. Automat. Contr., Vol. AC-26, pp. 911-917, 1981.

[4] K. V. Fernando and H. Nicholson, "Singular Perturbational Model Reduction in the Frequency Domain," IEEE Trans. Automat. Contr., Vol. AC-27, pp. 969-970, 1982.

THE FIXED POINT NUMERICAL SOLUTION OF ALGEBRAIC RICCATI EQUATION OF SINGULARLY PERTURBED SYSTEMS

ZORAN GAJIC
Rutgers University, Box 909, Department of Electrical Engineering
Piscataway, NJ 08854

Abstract. In this paper a fixed point algorithm for solving algebraic Riccati equation of singularly perturbed systems is considered. It is shown that each iteration step in the proposed algorithm improves the accuracy by an order of magnitude, i.e., the accuracy of $O(\varepsilon^k)$ can be obtained by doing only k-1 iterations. On the other hand, only low-order systems are involved in algebraic manipulations and no analicity requirements are imposed on the system coefficients.

Numerical solution of well-known Riccati equation

$$PA + A^TP + Q - PSP = 0, \quad Q = Q^T, \quad S = S^T \tag{1}$$

has been studied by many researchers /1/. For singularly perturbed systems, /2/, Riccati equation has a parametrized form

$$P(\varepsilon)A(\varepsilon) + A^T(\varepsilon)P(\varepsilon) + Q - P(\varepsilon)S(\varepsilon)P(\varepsilon) = 0 \tag{2}$$

where ε is a small positive parameter and

$$A(\varepsilon) = \begin{bmatrix} A_1 & A_2 \\ \frac{A_3}{\varepsilon} & \frac{A_4}{\varepsilon} \end{bmatrix}, \quad S(\varepsilon) = B(\varepsilon)R^{-1}B^T(\varepsilon) \qquad R > 0$$

$$B(\varepsilon) = \begin{bmatrix} B_1 \\ \frac{B_2}{\varepsilon} \end{bmatrix}, \quad Q(\varepsilon) = \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix}, \quad P(\varepsilon) = \begin{bmatrix} P_1 & \varepsilon P_2 \\ \varepsilon P_2^T & \varepsilon P_3 \end{bmatrix}$$

with dimensions compatible to $\dim P_1 = (n \times n)$ and $\dim P_2 = (m \times m)$.

Using standard techniques /1/, for solving (2) can be inappropriate. When ε is very small (Jacobian close to being singular) we are faced with ill-defined numerical problem. One way to overcome this situation is a power series expansion method. But presence of small parameter ε gives us the other way of solving this Riccati equation without requiring analicity conditions and producing numerically very efficient algorithm.

Pluging $\varepsilon = 0$ in (2) we get zero order solution by solving two low-order Riccati equations /3/. Zero order solution $\underline{P}$ is $O(\varepsilon)$ close to the exact one

$$P - \underline{P} = \varepsilon E \tag{3}$$

or by using compatible partition

$$\begin{bmatrix} \varepsilon E_1 & \varepsilon^2 E_2 \\ \varepsilon^2 E_2^T & \varepsilon^2 E_3 \end{bmatrix} = \begin{bmatrix} P_1 - \underline{P}_1 & \varepsilon (P_2 - \underline{P}_2) \\ \varepsilon (P_2 - \underline{P}_2)^T & \varepsilon (P_3 - \underline{P}_3) \end{bmatrix} \tag{4}$$

This work was supported by the National Science Fondation under Grant No ECF-8205337 with Michigan State University

where E is the error term. Clearly an $O(\varepsilon^k)$ approximation of E will produce $O(\varepsilon^{k+1})$ approximation of the sought solution P. That is why we are interested in finding a convenient form for the error term and an appropriate algorithm for its solution.

After doing some algebra we get the following set of error equations

$$\begin{aligned} E_3\underline{D}_3 + \underline{D}_3^T E_3 &= \gamma_3 + \varepsilon h_3(E_2,E_3,\varepsilon) \\ E_2\underline{D}_3 + E_1\underline{D}_{21} + \underline{D}_{22}^T E_3 &= \gamma_2 + \varepsilon h_2(E_1,E_2,E_3,\varepsilon) \\ E_1\underline{D}_1 + \underline{D}_1^T E_1 &= \gamma_1 + \varepsilon h_1(E_1,E_2,E_3,\varepsilon) \end{aligned} \tag{5}$$

where

$$S_1 = B_1R^{-1}B_1^T \quad , \quad S_2 = B_2R^{-1}B_2^T \quad , \quad S = B_1R^{-1}B_2^T$$

$$\underline{D}_3 = (A_4 - S_2\underline{P}_3), \quad \underline{D}_{21} = (A_2 - S\underline{P}_3), \quad \underline{D}_{22} = (A_3 - S^T\underline{P}_1 - S_2\underline{P}_2^T)$$

$$\underline{D}_1 = (A_1 - S_1\underline{P}_1 - S\underline{P}_2^T) - \underline{D}_{21}\underline{D}_3^{-1}\underline{D}_{22}$$

Note that under bar denotes matrices evaluated at $\varepsilon = 0$, and that $\gamma_1, \gamma_2, \gamma_3$ are known constants. Equations (5) have a very nice form since all cross coupling terms and all nonlinear terms are multiplied by a small parameter ε. This suggests that a fixed point algorithm can be efficient for their solution. Let us propose the following algorithm

$$\begin{aligned} E_3^{(i+1)}\underline{D}_3 + \underline{D}_3^T E_3^{(i+1)} &= \gamma_3 + \varepsilon h_3(E_2^{(i)}, E_3^{(i)}, \varepsilon) \\ E_2^{(i+1)}\underline{D}_3 + E_1^{(i+1)}\underline{D}_{21} + \underline{D}_{22}^T E_3^{(i+1)} &= \gamma_2 + \varepsilon h_2(E_1^{(i)},E_2^{(i)},E_3^{(i)},\varepsilon) \\ E_1^{(i+1)}\underline{D}_1 + \underline{D}_1^T E_1^{(i+1)} &= \gamma_1 + \varepsilon h_1(E_1^{(i)},E_2^{(i)},E_3^{(i)},\varepsilon) \end{aligned} \tag{6}$$

with starting points $E_1^{(0)} = E_2^{(0)} = E_3^{(0)} = 0$. Note that $\underline{D}_1$ and $\underline{D}_3$ are stable matrices. Then the following theorem holds.

THEOREM: Given algorithm (6) converges to the exact solution E with the rate of convergence of $O(\varepsilon)$, i.e.

$$\| E - E^{(i)} \| = O(\varepsilon^i) \qquad i = 0,1,2,.....$$

Proof of this theorem can be found in /4/.

ACKNOWLEDGMENTS - The author wishes to express his gratitude to Prof. H. Khalil for his suggestions and discussion in the course of this work.

REFERENCES:

/1/ M. Jamshidi, J. Large Scale Systems, pp. 167-192, 1980.
/2/ P. Kokotovic, R. O'Malley jr., and P. Sannuti, Automatica, Vol. 12, pp. 123-132, 1976.
/3/ J. Chow and P. Kokotovic, IEEE Trans. Aut. Cont. AC-21, 1976.
/4/ Z. Gajic, Ph.D. Thesis, Michigan State University, 1984.

A COMPOSITE CONTROL STRATEGY FOR A WEAKLY ELASTIC ROBOT *

ANTONIO FICOLA, RICCARDO MARINO, SALVATORE NICOSIA
Dipartimento di Ingegneria Elettronica - II Università di Roma "Tor Vergata"
Via O. Raimondo, 00173 - Roma (Italy)

ABSTRACT

A composite additive control strategy based on singular perturbation theory is proposed and tested for a three weakly elastic rotary joint robot.

Model Reference Adaptive Control (MRAC) techniques are used as far as the slow dynamics is concerned; we propose Variable Structure Control Systems (VSCS) in order to drive the fast state variables, which are the elastic forces and their time derivatives, to a specified submanifold in the state space.

Simulation tests are reported with reference to a significant trajectory: a straight line with variable velocity. They show that oscillations, which occur when only the MRAC algorithm is used, are removed when the proposed control strategy is adopted; preliminary simulation tests also show that the composite control is effective when the amount of elasticity is significantly increased, and that the effect of load changes on control performances is very small.

1. MAIN RESULTS

In the dynamic control of robots elasticity is very often neglected or treated as a disturbance or a modelling inaccuracy. In the first simulation test we present (Fig. 1) it is shown that a very satisfactory MRAC technique [6] for the control of a three rigid joint robot cannot prevent the structure from elastic oscillations at the end of a typical trajectory. This situation and the fact that control sampling frequencies can excite the natural frequencies of the structure even in presence of small amounts of elasticity call for a more accurate dynamic model and more complex control techniques.

In the following we consider flexible joints and maintain the reasonable assumption of small elasticity. We refer to the dynamic model of a three rotary joint robot arm with three degrees of freedom presented in [9] and obtained by using symbolic manipulation systems. It has been shown in [2] and [5] that the model can be written as (see the Appendix for details)

$$\begin{cases} \dfrac{d^2 q_1}{dt^2} = f_1(q_1,\ q_1 + \varepsilon K^{-1} z,\ \dot{q}_1,\ \dot{q}_1 + \varepsilon K^{-1}\dot{z}) + \\ \qquad + H_1(q_1,\ q_1 + \varepsilon K^{-1} z)\ z + G_1(q_1,\ q_1 + \varepsilon K^{-1} z)\ u(t) \\ \dfrac{d^2 z}{d\tau^2} = \varepsilon \dfrac{d^2 z}{dt^2} = f_2(q_1,\ q_1 + \varepsilon K^{-1} z,\ \dot{q}_1,\ \dot{q}_1 + \varepsilon K^{-1}\dot{z}) + \\ \qquad + H_2(q_1,\ q_1 + \varepsilon K^{-1} z)\ z + G_2(q_1,\ q_1 + \varepsilon K^{-1} z)\ u(t) \end{cases} \tag{1}$$

* This work was partly supported by MPI (fondi 40%).

where $q_1 = (q^1, q^2, q^3)$ is the vector of displacements of the rotors of joint motors with respect to the preceding links; $q_2 = (q^4, q^5, q^6)$ is the vector of displacements (due to elasticity) of the links with respect to the preceding motors; $u = (u_1, u_2, u_3)$ are the input torques delivered by the motors; $z = (z^1, z^2, z^3) = (\frac{1}{\varepsilon}\bar{k}_1 q_4, \frac{1}{\varepsilon}\bar{k}_2 q_5, \frac{1}{\varepsilon}\bar{q}_3 q_6)$ is the vector of elastic forces (we also write with a short hand notation $q_2 = \varepsilon K^{-1} z$); $\tau = t/\sqrt{\varepsilon}$ denotes the fast time scale when ε is small enough. We recall that $H_2(q_1, q_1)$ is nonsingular for all $(q_1, z) \in \mathbb{R}^6$.

As usual in singular perturbation theory we can formally write two reduced order systems:
the slow reduced system

$$\begin{cases} \dfrac{d^2 q_1}{dt^2} = f_1(q_1, q_1, \dot{q}_1, \dot{q}_1) + H_1(q_1, q_1)\, z + G_1(q_1, q_1)\, u(t) \\ z = - H_2^{-1}(q_1, q_1)\, f_2(q_1, q_1, \dot{q}_1, \dot{q}_1) - H_2^{-1}(q_1, q_1)\, G_2(q_1, q_1)\, u(t) \end{cases} \tag{2}$$

and the fast reduced system

$$\frac{d^2 z}{d\tau^2} = f_2(\hat{q}_1, \hat{q}_1, \hat{\dot{q}}_1, \hat{\dot{q}}_1) + H_2(\hat{q}_1, \hat{q}_1)\, z + G_2(\hat{q}_1, \hat{q}_1)\, u(t) \tag{3}$$

where $(\hat{q}_1, \hat{\dot{q}}_1)$ are parameters in the sense that $d\hat{q}/d\tau = 0$, $d\hat{\dot{q}}/d\tau = 0$. Let us restrict ourselves to composite state feedback control strategies of type

$$u = u_s(q_1, \dot{q}_1) + u_f\left\{ z - \hat{z},\ \frac{d}{d\tau}(z - \hat{\dot{z}}) \right\} \qquad u_f(0, 0) = 0 \tag{4}$$

where

$$\begin{cases} \hat{z} = - H_2^{-1}(\hat{q}_1, \hat{\dot{q}}_1)\, f_2(\hat{q}_1, \hat{\dot{q}}_1, \hat{q}_1, \hat{\dot{q}}_1) - H_2^{-1}(\hat{q}_1, \hat{q}_1)\, G_2(\hat{q}_1, \hat{q}_1)\, u_s(\hat{q}_1, \hat{\dot{q}}_1) \\ \left.\dfrac{\hat{dz}}{dt}\right|_{\substack{q_1 = \hat{q}_1 \\ \dot{q}_1 = \hat{\dot{q}}_1}} = - \dfrac{d}{dt}\left(H_2^{-1} f_2 - H_2^{-1} G_2 u_s\right) \Bigg|_{\substack{q_1 = \hat{q}_1 \\ \dot{q}_1 = \hat{\dot{q}}_1}} \end{cases} \tag{5}$$

Then the slow reduced system becomes

$$\frac{d^2 q_1}{dt^2} = f_R(q_1, \dot{q}_1) + G_R(q_1)\, u_s(q_1, \dot{q}_1) \tag{6}$$

where

$$f_R(q_1, \dot{q}_1) = f_1(q_1, \dot{q}_1, q_1, \dot{q}_1) - H_1(q_1, q_1)\, H_2^{-1}(q_1, q_1)\, f_2(q_1, \dot{q}_1, q_1, \dot{q}_1)$$

$$G_R(q_1) = G_1(q_1, q_1) - H_2^{-1}(q_1, q_1)\, G_2(q_1, q_1)$$

and the fast reduced system becomes (see the Appendix)

$$\frac{d^2(z-\hat{z})}{d\tau^2} = H_V(\hat{q}_1, \dot{\hat{q}}_1)\,(z - \hat{z}) + G_V(\hat{q}_1)\, u_f\left(z - \hat{z}, \frac{d}{d\tau}(z - \hat{z})\right) \qquad (7)$$

System (6) coincides with the model of the same robot we are considering when joints are stiff. In this case q_1 denotes the displacements among subsequent links. On the other hand the fast reduced system (7) has the advantage of being linear ($(\hat{q}_1, \dot{\hat{q}})$ act as parameters) and controllable for all parameters $\hat{q}_1$, $\dot{\hat{q}}_1$ (cfr. [5]). Tikonov's theorem applies whenever fast controls which stabilize (7) to $\left(\hat{z}, \left(\frac{d\hat{z}}{dt}\right)\right)$ are employed. In this case the orbits of the original system (1) approximate in the limit $\varepsilon \to 0$ the orbits of the reduced systems (6) and (7). Thus we can use equations (6) and (7) for design purposes in the case of weakly elastic joints. Note that the resulting control strategy is a decentralized one. System (6) can be transformed by state feedback into a linear and controllable system. By virtue of this special property many satisfactory control laws are available ([3],[4],[6]).

We choose to use the model reference adaptive control algorithm since it was found very robust in extensive simulation tests [6]. Its performance is evaluated by simulation when it is applied to the weakly elastic joint robot model (1) (see the Appendix for the exact values of parameters) in order to track a straight line with variable velocity. Oscillations arise mainly because the elastic forces and their time derivatives are continuously driven away from their equilibrium points which vary along the trajectory and are approximately given by (5). In fact the explicit expressions (5) constitute the main tool which allows a stabilization of elastic oscillations. Exploiting diagonal dominance of matrix $G_V(\hat{q}_1)$ we propose a variable structure control system (cfr. [7] p. 101) for the stabilization of (7) to the origin; denote $\tilde{z} = z - \hat{z}$, $\tilde{z}' = \frac{d}{d\tau}(z - \hat{z})$;

$$u_{f1} = \begin{cases} u_{f1}^{+} & \text{if} \quad \tilde{z}_1' + c_1 \tilde{z}_1 > 0 \\ u_{f1}^{-} & \text{if} \quad \tilde{z}_1' + c_1 \tilde{z}_1 < 0 \end{cases}$$

where $c_1 > 0$ and u_{f1}^{+}, u_{f1}^{-} must satisfy the following inequalities

$$\operatorname{sgn}(h_{41})\, u_{f1}^{+} < -\frac{h_{44}\tilde{z}_1 + c_1 \tilde{z}_1'}{|h_{41}|}$$

$$\operatorname{sgn}(h_{41})\, u_{f1}^{-} > -\frac{h_{44}\tilde{z}_1 + c_1 \tilde{z}_1'}{|h_{41}|}$$

$$u_{f3} = \begin{cases} u_{f3}^{+} & \text{if} \quad \tilde{z}_3' + c_3 \tilde{z}_3 > 0 \\ u_{f3}^{-} & \text{if} \quad \tilde{z}_3' + c_3 \tilde{z}_3 < 0 \end{cases}$$

where $c_3 > 0$ and u_{f3}^{+} , u_{f3}^{-} must satisfy

$$\text{sgn}\,(h_{63})\, u_{f3}^{+} < - \frac{h_{65}\,\tilde{z}_2 + h_{66}\,\tilde{z}_3 + c_3\,\tilde{z}_3'}{|\,h_{63}\,|}$$

$$\text{sgn}\,(h_{63})\, u_{f3}^{-} > - \frac{h_{65}\,\tilde{z}_2 + h_{66}\,\tilde{z}_3 + c_3\,\tilde{z}_3'}{|\,h_{63}\,|}$$

$$u_{f2} = \begin{cases} u_{f2}^{+} & \text{if} \quad \tilde{z}_2' + c_2 \tilde{z}_2 > 0 \\ u_{f2}^{-} & \text{if} \quad \tilde{z}_2' + c_2 \tilde{z}_2 < 0 \end{cases}$$

where $c_2 > 0$, u_{f2}^{+} , u_{f2}^{-} must satisfy

$$\text{sgn}\,(h_{52})\, u_{f2}^{+} < - \frac{c_2\,\tilde{z}_2' + h_{55}\,\tilde{z}_2 + h_{56}\,\tilde{z}_3 + h_{53}\,u_{f3}^{eq}}{|\,h_{52}\,|}$$

$$\text{sgn}\,(h_{52})\, u_{f2}^{-} > - \frac{c_2\,\tilde{z}_2' + h_{55}\,\tilde{z}_2 + h_{56}\,\tilde{z}_3 + h_{53}\,u_{f3}^{eq}}{|\,h_{52}\,|}$$

and

$$u_{f3}^{eq} = - \frac{h_{65}\,\tilde{z}_2 + h_{66}\,\tilde{z}_3 + c_3\,\tilde{z}_3'}{|\,h_{63}\,|}$$

Fortunately there is no need of modulating the controls since $\text{sgn}\,(h_{41}) < 0$, $\text{sgn}\,(h_{63}) < 0$ and $\text{sgn}\,(h_{52}) < 0$.

2. CONCLUSIONS

Extensive simulations and actual experiences on a robot arm prototype of a leading italian industry show that oscillations arise when control algorithms designed under the hypothesis of stiff joints are employed in presence of certain amounts of elasticity. This happens even in the case of a single flexible joint.

A singular perturbation analysis makes it clear that oscillations arise because elastic displacements and their time derivatives are driven away from the instantaneous equilibrium reference which is not known. We provide an approximate

closed form expression for the equilibrium reference to which the fast dynamics is stabilized to by VSCS techniques.

Since the components of the composite control law proposed are very robust in nature it is expected that control performances remain acceptable even in case of load changes or larger amounts of elasticity: preliminary simulation tests confirm this conjecture.

3. APPENDIX

$$H_V = \begin{pmatrix} h_{44} & 0 & 0 \\ 0 & h_{55} & h_{56} \\ 0 & h_{65} & h_{66} \end{pmatrix}$$

$$G_V = \begin{pmatrix} h_{41} & 0 & 0 \\ 0 & h_{52} & h_{53} \\ 0 & 0 & h_{63} \end{pmatrix}$$

where

$$h_{41} = 1/a_1$$

$$h_{52} = 1/a_2$$

$$h_{53} = a_3\, a_4/\det$$

$$h_{63} = a_3 \left(\frac{b_3}{2} \cos q_3 + b_8 \right)/\det$$

$$h_{44} = -1/a_1 - 1/\left(b_2 \cos^2 q_2 + b_3 \cos (q_2 + q_5) \right.$$

$$\left. \cos (q_2 + q_3) + b_5 \cos^2 (q_2 + q_3) + b_6 \right)$$

$$h_{55} = -1/a_3 - a_3\, a_4/\det$$

$$h_{56} = \left(a_3 \left(\frac{b_3}{2} \cos q_3 + b_8 \right) - a_3\, a_4 \right)/\det$$

$$h_{65} = \left(a_3 \left(\frac{b_3}{2} \cos q_3 + b_8 \right) - a_3\, a_4 \right)/\det$$

$$h_{66} = \left(a_3^2 + 2\, a_3 \left(\frac{b_3}{2} \cos q_3 + b_8 \right) - a_3 (b_3 \cos q_3 + b_7) \right)/\det$$

where $\det = a_3 \left(a_4 (b_3 \cos q_3 + b_7) - \left(\frac{b_3}{2} \cos q_3 + b_8 \right)^2 \right)$ and

H_{21} is nonsingular in $\mathbb{R}^{12}$.

4. REFERENCES

[1] V.R. Saksena, J. O'Reilly, P.V. Kokotovic - Singular perturbations and time--scale methods in control theory - *Survey 1976-1983, AUTOMATICA* - Vol. 20, n. 3, pp. 273-293, 1984.

[2] A. Ficola, R. Marino, S. Nicosia - A singular perturbation approach to the dynamic control of elastic robots - *21st Annual Allerton Conference on Communication, Control and Computing* - Monticello, Illinois, Oct. 1983.

[3] E. Freund - Fast nonlinear control with arbitrary pole placement for industrial robots and manipulators - *Int. J. Robotics Research 1* - 1, 65-78, 1982.

[4] S. Nicosia, F. Nicolò, D. Lentini - Dynamical control of industrial robots with elastic and dissipative joints - *8th IFAC World Congress* - Kyoto, Aug. 1981.

[5] R. Marino, S. Nicosia - On the feedback control of industrial robots with elastic joints: a singular perturbation approach - *Report R84.01* - Dipartimento di Ingegneria Elettronica, II Università di Roma.

[6] S. Nicosia, P. Tomei - Model reference adaptive control algorithms for industrial robots - *AUTOMATICA* - to appear in Sept. 1984 issue.

[7] V.I. Utkin - Sliding modes and their applications in variable structure systems - *MIR Publisher* - Moscow, 1978.

[8] N. Fenichel - Geometric singular perturbation theory for ordinary differential equations - *Journal of Differential Equations* - Vol. 31, pp. 53-98, 1979.

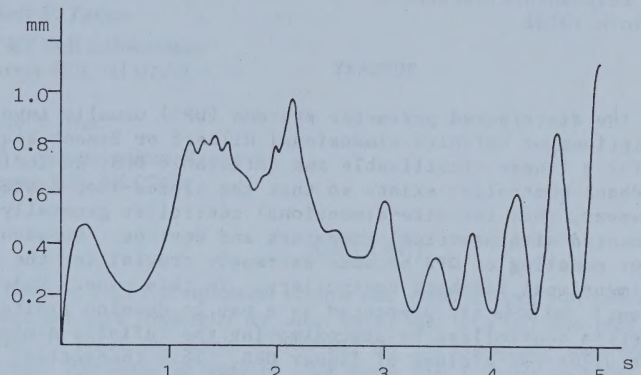


Fig.1

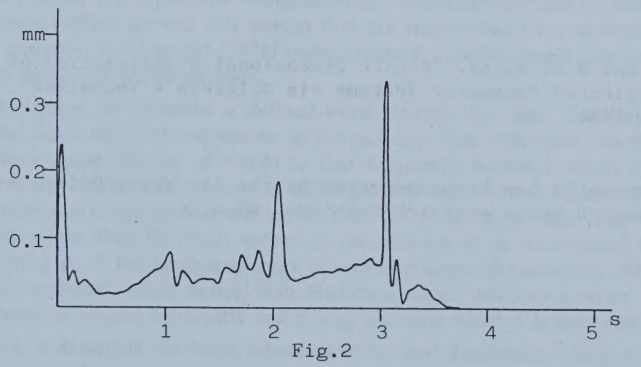


Fig.2

REDUCED ORDER MODELING AND FINITE DIMENSIONAL STABILIZATION OF LINEAR DISTRIBUTED PARAMETER SYSTEMS VIA GALERKIN'S TECHNIQUE*

J.G. HSIEH and M.J. BALAS
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12181

SUMMARY

Control of the distributed parameter systems (DPS) usually involves dynamical descriptions on infinite-dimensional Hilbert or Banach spaces of functions. For a linear stabilizable and detectable DPS, an infinite-dimensional feedback controller exists so that the closed-loop system is stable [1]. However, this infinite-dimensional controller generally cannot be implemented with practical computers and devices. Consequently, the reduced order modeling of DPS becomes extremely crucial for the design of the finite-dimensional feedback controllers. In this paper, Galerkin (or finite element) methods are presented as a way to develop finite-dimensional Galerkin controllers by approximating the infinite-dimensional stabilizing controller for a class of linear DPS. This theoretical investigation will give a very simple sufficient condition, i.e. the test of nonsingularity of a matrix, for a finite-dimensional Galerkin feedback controller to perform as well as the infinite-dimensional stabilizing controller [2].

REFERENCES

1. M. Balas, "Toward a (More) Practical Control Theory for Distributed Systems", in Control and Dynamic Systems: Advances in Theory and Applications, Vol. 18 (C. Leondes, ed.), Academic Press, New York, 1982.

2. J.G. Hsieh and M.J. Balas, "Finite Dimensional Stabilization of Linear Distributed Parameter Systems via Galerkin's Technique", (preprint), 1984.

*The research reported has been supported by the Air Force Office of Scientific Research under research Grant No. 83-0124.

A LOCALLY ADAPTIVE DATA COMPRESSION SCHEME

Jon Louis Bentley
Daniel D. Sleator
Robert E. Tarjan
AT&T Bell Laboratories
Murray Hill, NJ 07974

Victor K. Wei
Bell Communications Research
Murray Hill, NJ 07974

ABSTRACT

We describe a data compression scheme that exploits locality of reference, which occurs when words are used frequently for short bursts and then fall into long periods of disuse. The scheme is based on simple a heuristic for self-organizing sequential search and on variable-length encodings of integers. We prove that it never performs much worse than Huffman codes and can perform substantially better; experiments on real files show that its performance is usually quite close to Huffman codes. This scheme has many implementation advantages: it is simple, allows fast encoding and decoding, and requires only one pass over the data to be compressed (Huffman codes require two passes).

1. INTRODUCTION

Data compression schemes can be categorized by the unit of data they transmit. Huffman [1951] codes are typical of "defined-word" schemes: the context defines sequences of input symbols (which we will call words) that are transmitted by a variable-length code. At the other extreme, Ziv-Lempel [1978] codes transmit variable-length sequences of input symbols, often using a fixed-length code.

In this paper we describe a defined-word scheme that uses a technique from another domain that deals with defined words: self-organizing data structures. In that context we are to maintain a sequential list of words so that frequently accessed words are near the front. Our data compression scheme uses a self-organized list as an auxiliary data structure, and employs short encodings to transmit words near the front of the list. The scheme never performs much worse than Huffman codes. If the message to be transmitted exhibits locality of reference (that is, if the local frequency of words changes dramatically within the message), the scheme performs much better than Huffman codes, because a word will have a short encoding when it is used frequently and a long encoding when it is used rarely.

Section 2 describes the basic scheme and several dimensions along which it may vary. Mathematical analyses of the performance of the scheme are given in Section 3 (and an appendix), and experimental evidence is presented in Section 4. Section 5 mentions implementation considerations, and conclusions are offered in Section 6.

2. THE THEME AND SOME VARIATIONS

We will illustrate our scheme by compressing simple "telegraph" messages of words consisting of upper case letters separated by single spaces and terminated by a final "•". For concreteness, we will transmit the message

THE CAR ON THE LEFT HIT THE CAR I LEFT •

Sender and receiver maintain identical word lists using the "Move-to-Front" heuristic: after a

word is used it is deleted from its current position and moved to the front of the list; this attempts to ensure that common words appear near the front of the list.

The list is initially empty. To transmit the word W, the sender looks it up in the list. If it is present in position I, the sender transmits I, which the receiver decodes by writing the I^{th} element in the list; both then move W to the front of their respective lists, shifting words $1..I-1$ to positions $2..I$. If W is not in the list of N words, the sender reacts as though it were in the $N+1^{st}$ position and sends the integer $N+1$ followed by the word W (which the receiver expects because $N+1$ is greater than the size of the current list); both sender and receiver then move W to the front of their lists. So after transmitting the first three words of the above message, both parties have the identical lists

ON CAR THE

The next word, THE, is encoded by the integer 3. The entire message is encoded as

1 THE 2 CAR 3 ON 3 4 LEFT 5 HIT 3 5 6 I 5 •

Each word is transmitted as a string of letters just once; subsequent occurrences are encoded by an integer.

This trivial example illustrates the most important property of our scheme: if a word has been recently used then it will be near the front of the list and therefore have a short decimal encoding. Because the integer I requires roughly $\log_{10} I$ characters to encode, frequent words are transmitted with few characters. There are, however, many variations on the basic idea.

Lexical Analysis. English text may be divided into "words" in many ways. A simple scheme might classify each character as a word, while a more complex scheme could find true English words, together with capitalization information. Transmitting program text, executable object code, or digitally encoded pictures demands a more subtle definition of words.

List Organization Discipline. Bentley and McGeoch [1982] and Sleator and Tarjan [1984] refer to many self-organization heuristics beyond Move-to-Front. Transpose, for instance, moves the accessed element one closer to the front; it is an instance of the Move-Ahead-k heuristic with $k=1$.

List Length. The above example assumed an infinite list; the scheme may also be implemented with fixed-size lists. The Move-to-Front scheme with a finite list induces a Least-Recently-Used discipline of discarding words from the list (which, in that context, may be viewed as a word cache).

Encoding List Position. Position in a finite list can be encoded with a fixed-length binary code, but the scheme is usually more effective if used with a variable-length code. If the data is to be transmitted as it is read, the variable-length prefix encodings of integers described by Elias [1975] and by Bentley and Yao [1977] provide suitable encodings; these will be discussed in detail later. If, on the other hand, the system can make two passes over the data then the first pass can count the number of times each list position is accessed and a the second pass can encode them using a Huffman code.

Transmission of New Words. This is a classical problem in information theory.

We will see several combinations of these choices in the next sections.

3. THEORETICAL ANALYSIS OF PERFORMANCE

In this section we show that the Move-to-Front scheme is sometimes much better than Huffman codes, but it can never be much worse. Here we summarize our theoretical results; the appendix contains a more complete analysis.

A simple example shows that the Move-to-Front scheme can be much better than any static encoding scheme. Consider the sequence formed by repeating each of n words n times: $1^n 2^n \cdots n^n$. A static Huffman code uses roughly $\log n$ bits per word sent*, whereas most

*All logarithms in this paper without an explicit base are binary.

implementations of the Move-to-Front scheme use only a small constant number of bits per word.

To analyze the Move-to-Front scheme we need to specify a particular encoding of the integers. One method is to prefix the binary representation of the integer $i \geq 1$ with $\lfloor \log i \rfloor$ 0's. This yields a prefix code since the total length of a codeword is exactly one plus twice the number of 0's in the prefix. Once the length is known the boundary between codewords can be found. This method encodes i with $1+2\lfloor \log i \rfloor$ bits.

We will analyze the scheme in which the size of the MTF cache is equal to the number of different words to be sent, and the cache is initialized with all the words in their order of occurrence in the sequence of words to be sent. This is the same scheme that was outlined in section 2, except that we ignore the cost of sending the raw words (we ignore this cost in both of the schemes being compared). Let $\rho_{MF}(X)$ be average number of bits per word used to compress a sequence X with the MTF cache as described above. Let $\rho_H(X)$ be the analogous quantity for a static Huffman code. With the above encoding of the integers we have

$$\rho_{MF}(X) \leq 2\rho_H(X)+1.$$

We will now sketch an intuitive proof of this theorem; a proof of a stronger result appears in the appendix. Suppose in the sequence X of length N, the symbol a occurs N_a times. The distance a can move from the front of the list between successive occurrences of a is bounded by the number of other accesses between these occurrences. The average size of these gaps between successive accesses to a is about N/N_a. If a were at its average position whenever it occurred in the sequence, then the number of bits needed to transmit each occurrence would be $2 \log(N/N_a)+1$ bits. Although a is not always at this position in the list, it turns out (because of the convexity of the log function) that $2 \log(N/N_a)+1$ is an upper bound on the average cost of transmitting an a. Since the cost of transmitting a with an optimum prefix code is at least $\log(N/N_a)$, the result follows.

By using more sophisticated encodings of integers, stronger results can be proved. For example:

$$\rho_{MF}(X) \leq 1+\rho_H(X)+2 \log(1+\rho_H(X)).$$

These results compare two schemes that differ in two important ways. The Move-to-Front scheme is dynamic, (the encoding of a word may change with time), while Huffman codes are static (they are fixed in advance). This should, of course, give Move-to-Front an advantage. On the other hand, the MTF scheme works on line (a word can be transmitted immediately when it first appears), while Huffman codes require a pass over all the data before anything can be sent.

We do not know to compare our scheme with other dynamic schemes. In particular we would like to know how our scheme compares to a dynamic Huffman scheme developed by Gallager [1978]. This scheme maintains a dynamically changing weight for each word. At each occurrence of a word its weight is increased by one, and periodically all the weights are multiplied by a factor less than 1. To encode each word an optimum Huffman code is used for the current distribution of weights.

A theorem in the appendix compares the performance of the Move-to-Front method when compressing a discrete memoryless source with the entropy of the source. This result shows that the MTF scheme is asymptotically optimum (see Elias [1978]) because:

$$\frac{<\rho_{MF}(X)>}{H(s)} \rightarrow 1 \text{ as } H(s) \rightarrow \infty$$

where $H(s)$ is the entropy of the source and $<\rho_{MF}(X)>$ is the expected number of bits per word sent of the MTF scheme over all sequences.

4. EXPERIMENTAL ANALYSIS OF PERFORMANCE

To gain further insight into the performance of our scheme we implemented prototypes of the following three compression algorithms.

- *Byte-level Huffman Code.* A Huffman code is used on individual bytes. We did not charge for transmitting the Huffman tree, which requires less than 200 bytes.
- *Word-level Huffman Code.* Words were defined as longest sequences of alphanumeric and nonalphanumeric characters. This divides the input stream into two disjoint classes, which we therefore compressed separately; the decoder knows to alternate the two classes when decoding. No case information was recorded; "the" and "The" were thus treated as distinct words. The words were transmitted using a byte-level Huffman code (as above, without charge for transmitting the tree).
- *MTF-Cache.* Many attributes of this scheme are the same as for the word-level Huffman code, including word definition, two word sets, lack of case information, and transmission of words. Because there are two word sets, the 8-element MTF cache stores 16 words (8 alphanumeric and 8 nonalphanumeric). For ease of implementing the prototype, we encoded the position in the list by a Huffman code, which implies that an implementation would have to make two passes over the data.

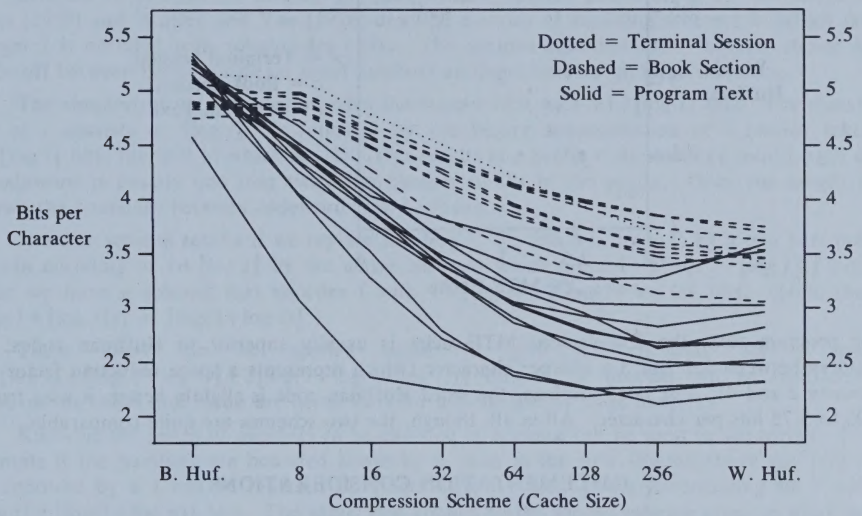
None of the implementations actually compresses and restores data; rather, they measure the efficiency of the various approaches. We view this work only as a preliminary experiment to demonstrate the plausibility of the scheme and to gain insight into its behavior.

The results of the experiments are presented in the following table. The numbers show the bits per character used by each scheme (the original encoding uses 8 bits per character, but that can be trivially reduced to 7).

File Type	Size in Bytes	Huffman Code		Move-to-Front Cache					
		Byte	Word	8	16	32	64	128	256
Program	20593	5.285	2.958	4.253	3.785	3.385	3.049	2.917	2.822
Text	19118	4.576	2.321	3.071	2.356	2.245	2.196	2.183	2.233
(C)	17160	5.248	3.026	4.672	4.147	3.630	3.217	3.033	3.019
	17470	5.210	3.520	4.510	4.028	3.814	3.637	3.413	3.379
	18441	5.288	3.585	4.505	4.088	3.645	3.388	3.263	3.270
	16282	5.150	2.316	4.044	3.517	2.784	2.368	2.253	2.271
	23225	5.321	2.219	4.025	3.269	2.711	2.408	2.269	2.113
(Pascal)	31930	5.102	2.467	4.468	3.892	3.312	2.856	2.661	2.501
	19144	5.267	2.678	4.424	3.902	3.392	3.035	2.801	2.642
	14673	5.124	2.521	4.323	3.750	3.327	2.906	2.580	2.454
	8535	5.082	2.662	4.452	3.968	3.446	3.022	2.778	2.675
	12833	5.349	2.822	4.343	3.886	3.478	3.177	2.945	2.627
Terminal Session	142762	5.168	3.412	5.101	4.783	4.504	4.212	3.929	3.685
Book	20616	4.832	3.460	4.826	4.551	4.274	3.958	3.735	3.559
Section	18225	4.748	3.374	4.807	4.471	4.164	3.879	3.632	3.471
	22360	4.978	3.698	4.985	4.695	4.424	4.140	3.992	3.852
	17471	4.878	3.747	4.880	4.629	4.382	4.125	3.967	3.847
	15709	4.883	3.617	4.971	4.638	4.363	4.108	3.921	3.739
	15104	4.856	3.453	4.837	4.486	4.136	3.840	3.637	3.509
	19113	5.073	3.530	4.923	4.579	4.239	3.934	3.729	3.594
	22346	4.902	3.054	4.723	4.270	3.819	3.547	3.313	3.134

The C and Pascal programs were written by several different programmers; the book sections (written by two sets of multiple authors) include TROFF formatting commands. The terminal session is the transcript of a several-hour session (we include it to underscore the point that the performance of the various schemes is quite dependent on the context).

Most of the data in the table is represented in the following graph, in which each file is represented by a line.

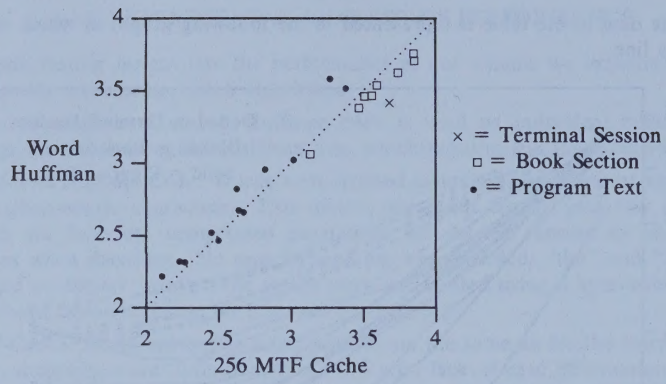


Each line represents (from left to right) the cost of the byte-level Huffman encoding, MTF Cache encodings of increasing sizes, and the word-level Huffman encoding.

This graph tells several stories. It is obvious that the three types of inputs have different characteristics; there is enough data on program text and book sections to draw plausible conclusions. Byte-level Huffman codes for the book sections are roughly as effective as the 8-element MTF cache; as the cache increases the MTF scheme shows steady improvement, but even at the 256-element cache it is still somewhat inferior to word-level Huffman codes. The relative ordering of the book sections is quite stable through all the encoding schemes.

The programs differ dramatically from the book sections. Byte-level Huffman codes are less effective for programs (presumably due to a larger character set), but the 8-element cache is already down to 4 to 4.5 bits per character (presumably due to strong locality of reference). The improvement with increasing cache size is erratic; some documents even exhibit non-monotonicity (retransmitting a few words costs less than sending longer codes for the words that are kept in the cache). The 256-element MTF cache is a little more effective than word-level Huffman codes. The outlying program contains a large table that represents legal hyphenations of English words, so one might expect it to be quite different.

The word-level Huffman encoding is compared to the 256-element MTF cache in the following graph.



For program text, the 256-element MTF-cache is usually superior to Huffman codes; it requires between 2.1 and 3.4 bits per character (which represents a space reduction factor of between 2 and 4). For book sections, the word Huffman code is slightly better; it uses from 3.05 to 3.75 bits per character. All in all, though, the two schemes are quite comparable.

5. IMPLEMENTATION CONSIDERATIONS

Space limitations prevent us from describing all of our ideas about how the scheme can be efficiently implemented. These results will be described fully in a future paper.

By using balanced trees the Move-to-Front scheme may be implemented in such a way that the time taken to update both the receiver's and sender's structures is proportional to the number of bits sent. Thompson [1984] has shown how to implement the scheme with a systolic array of cells, each of which stores one dictionary word.

6. CONCLUSIONS

We have described a simple data compression scheme and analyzed its performance both theoretically and experimentally. Both analyses suggest that the method may be useful in practice.

There are several problems left open by this paper: Is it possible to compare the Move-to-Front scheme with other dynamic schemes? Are there modifications of the scheme that are simpler to implement? Can other self-adjusting data structures be used for data compression?

APPENDIX: ANALYSIS

In order to analyze our scheme we need to have specific prefix codes for the integers. Elias [1975] and Bentley and Yao [1976] describe a series of encoding schemes in which the integer i is encoded with roughly $\log i$ bits. The various schemes differ in their choice of trade-off between performance on small numbers and performance on large numbers.

The simplest of the schemes encodes the integer $i \geq 1$ with $1+2\lfloor \log i \rfloor$ bits. The encoding of i consists of $\lfloor \log i \rfloor$ 0's followed by the binary representation of i (which takes $1+\lfloor \log i \rfloor$ bits, the first of which is a 1). This results in a prefix code since the total length of a codeword is exactly one plus twice the number of 0's in the prefix. Once the length is known the boundary between codewords can be found.

Another scheme results if we replace the $\lfloor \log i \rfloor$ 0's followed by a 1, by a two part prefix (an encoding of $1+\lfloor \log i \rfloor$ by the above scheme) which takes $1+2\lfloor \log(1+\lfloor \log i \rfloor) \rfloor$ bits. Thus we have a scheme that encodes i with $1+\lfloor \log i \rfloor + 2\lfloor \log(1+\log i) \rfloor$ bits. (Note that $\lfloor \log(1+\lfloor \log i \rfloor) \rfloor = \lfloor \log(1+\log i) \rfloor$.)

These ideas can be applied again to give an encoding for i with $1+\lfloor \log i \rfloor + \lfloor \log(1+\log i) \rfloor + 2\lfloor \log(1+\log(1+\log i)) \rfloor$ bits. This process can be continued, however the codes that result are better only for astronomically large numbers.

Knowing the range of numbers to be encoded in advance can be used to advantage. For example if the numbers are bounded above by n, then in the first scheme above the $\lfloor \log i \rfloor$ 0's followed by a 1 can be replaced by $\lfloor \log(1+\log n) \rfloor$ bits, giving an encoding for i with $\lfloor \log i \rfloor + \lfloor \log(1+\log n) \rfloor$ bits. The same idea applied to the second scheme gives an encoding of i in $\lfloor \log i \rfloor + \lfloor \log(1+\log i) \rfloor + \lfloor \log(1+(\log(1+\log n))) \rfloor$ bits.

For the following discussion we assume that an encoding of integers has been chosen, and that the number of bits needed to encode the integer i is at most $f(i)$, where $f(i)$ is a convex monotonically increasing function defined on real values of $i \geq 1$. For example if we choose the second scheme then we can let $f(i) = 1+\log i+2\log(1+\log i)$. We assume that the input stream has been partitioned into a sequence of dictionary words, which we will call symbols. Let the sequence of symbols be $X = x_1, x_2, \ldots, x_N$. The symbols are taken from a dictionary S of size n. Let $\rho_{MF}(X, f)$ be the average number of bits per symbol needed to transmit X by the Move-to-Front scheme using a code with codeword length function f. That is, $\rho_{MF}(X)$ is the total number of bits needed to transmit the sequence X divided by N. (From now on we omit the reference to f.) Let N_a be the number of occurrences of a symbol a in X. Then we have

Theorem 1:

$$\rho_{MF}(X) \leq \sum_{a \in S} \frac{N_a}{N} f\left(\frac{N}{N_a}\right)$$

Proof: Let $t_1, t_2, \cdots, t_{N_a}$ be the times when the N_a occurrences of the symbol a are sent. That is, $x_{t_i} = a$ and $t_i < t_{i+1}$. When a occurs at time t_1 its position in the list is at most t_1. Futhermore when a occurs at time t_i for $i > 1$ its position is at most $t_i - t_{i-1}$. Therefore the cost of transmitting the first a is at most $f(t_1)$, and the cost of transmitting the i^{th} a is at most $f(t_i - t_{i-1})$. If we let $R_a(X)$ be the total number of bits used to transmit the N_a occurrences of symbol a then

$$R_a(X) \leq f(t_1) + \sum_{i=2}^{N_a} f(t_i - t_{i-1}).$$

Noting the convexity of f and applying Jensen's inequality* we get

*Jensen's inequality says that if f is a convex function, $\{w_i\}$ is a set of positive real weights whose sum is 1, and $\{p_i\}$ is a set of points in the domain of f, then $\sum_i w_i f(p_i) \leq f(\sum_i w_i p_i)$ (Hardy, Littlewood, and Polya [1967], page 70).

$$R_a(X) \leq N_a f\left(\frac{1}{N_a}\left(t_1+\sum_{i=2}^{N_a} t_i - t_{i-1}\right)\right) = N_a f\left(\frac{t_{N_a}}{N_a}\right) \leq N_a f\left(\frac{N}{N_a}\right).$$

The equality follows from the fact that the terms $t_i - t_{i-1}$ telescope, and the second inequality follows from the fact that f is monotonically increasing. Summing over all $a \in S$ and dividing by N gives Theorem 1. Q.E.D.

By combining Theorem 1 with a particular encoding scheme we can relate the efficiency of the Move-to-Front compression method on a particular sequence to the value of the "empirical entropy" of the sequence. This entropy, H^*, is defined as follows.

$$H^*(X) \equiv \sum_{a \in S} -\frac{N_a}{N} \log \frac{N_a}{N}$$

Corollary 1:

$$\rho_{MF}(X) \leq 1 + H^*(X) + 2 \log(1 + H^*(X)).$$

Proof: We use the function f appropriate for the second scheme:

$$f(i) = 1 + \log i + 2 \log(1 + \log i).$$

Substituting this into Theorem 1 we get:

$$\rho_{MF}(X) \leq \sum_{a \in S} \frac{N_a}{N} + \sum_{a \in S} \frac{N_a}{N} \log \frac{N}{N_a} + \sum_{a \in S} \frac{N_a}{N} 2 \log\left(1 + \log \frac{N}{N_a}\right)$$

The value of the first sum is 1. The second sum is just $H^*(X)$.

Because log is a convex function and $\sum \frac{N_a}{N} = 1$ we can apply Jensen's inequality to the third summation to bound it by

$$2 \log\left[\sum_{a \in S} \frac{N_a}{N}\left(1 + \log \frac{N}{N_a}\right)\right].$$

The summation inside of the log is just $1 + H^*(X)$. Combining these pieces yields the desired result. Q.E.D.

We may now compare the performance of the Move-to-Front scheme with that of an optimum static prefix code for any particular sequence. One way of getting an optimum code for a particular sequence is to generate an optimum code for a source in which the probability of a symbol a occurring is $\frac{N_a}{N}$, which is just a Huffman code for this probability distribution. Let $\rho_H(X)$ be the average number of bits per symbol used by this code on the sequence X. A well known fact about an optimum static code is

$$H^*(X) \leq \rho_H(X) \leq H^*(X) + 1.$$

(See Chapter 3 of Gallager [1968].)

Substituting the left hand inequality into Corollary 1 gives us the following result.

$$\rho_{MF}(X) \leq 1 + \rho_H(X) + 2 \log(1 + \rho_H(X)).$$

Notice that the Move-to-Front scheme at its worst performs almost as well as a static optimum code, even though it has no advance knowledge of the sequence. Furthermore the Move-to-Front scheme will in fact do much better than the static optimum on certain types of sequences.

We have been able to evaluate the average performance of our scheme when compressing a sequence of symbols generated independently according to a fixed distribution. (This is called a discrete memoryless source.)

Theorem 2: If the symbols are generated by a discrete memoryless source in which $\text{Prob}\{x_t=a\} = P_a$, *then we have*

$$<\rho_{MF}(X)> \leq \sum_{a\in S} P_a f(\frac{1}{P_a}),$$

Where $<\cdot>$ *denotes expected value over all sequences of length N.*

Proof: Taking expected values on both sides of Theorem 1, we have

$$\begin{aligned}
<\rho_{MF}(X)> &\leq \sum_{a\in S} <\frac{N_a}{N} f(\frac{N}{N_a})> \\
&= \sum_{a\in S} \sum_{i=1}^{N} (\text{Prob}\{N_a=i\} \frac{i}{N} f(\frac{N}{i})) \\
&= \sum_{a\in S} \sum_{i=1}^{N} \binom{N}{i} P_a{}^i (1-P_a)^{N-i} \frac{i}{N} f(\frac{N}{i})) \\
&= \sum_{a\in S} P_a \sum_{i=1}^{N} \binom{N}{i} P_a{}^{i-1} (1-P_a)^{N-i} \frac{i}{N} f(\frac{N}{i})).
\end{aligned}$$

The next step is to pull f out of the inner summation using Jensen's inequality. To do this we must verify that $\sum_{i=1}^{N} \binom{N}{i} P_a{}^{i-1} (1-P_a)^{N-i} \frac{i}{N} = 1$. This follows immediately from the observation that $\binom{N}{i} \frac{i}{N} = \binom{N-1}{i-1}$, and the binomial theorem.

After applying Jensen's inequality we have

$$\begin{aligned}
<\rho_{MF}(X)> &\leq \sum_{a\in S} P_a f(\sum_{i=1}^{N} \binom{N}{i} P_a{}^{i-1} (1-P_a)^{N-i} \frac{N}{N}) \\
&= \sum_{a\in S} P_a f(\frac{[\sum_{i=0}^{N} \binom{N}{i} P_a{}^i (1-P_a)^{N-i}] - (1-P_a)^N}{P_a}).
\end{aligned}$$

The inner summation is equal to 1 by the binomial theorem.

$$\begin{aligned}
&= \sum_{a\in S} P_a f(\frac{1-(1-P_a)^N}{P_a}) \\
&\leq \sum_{a\in S} P_a f(\frac{1}{P_a}).
\end{aligned}$$

The last inequality follows from the monotonicity of f. Q.E.D.

Note: Theorem 2 holds for all values of $N \geq 1$.

Using Theorem 2 we can derive a corollary that bounds the expected performance of the Move-to-Front method in terms of the entropy of the source s. This is an expected-case version of Corollary 1. Let $H(s)$ denote the entropy of the source:

$$H(s) \equiv \sum_{a\in S} -P_a \log P_a$$

Corollary 2:

$$<\rho_{MF}(X)> \leq 1+H(s)+2\log(1+H(s)).$$

Where the average is taken over all sequences of length N.

This follows from Theorem 2 in much the same way that Corollary 1 follows from

Theorem 1. A substitution is made for f into Theorem 2, then the three summations are bounded using Jensen's inequality.

Corollaries similar to 1 and 2 above can be proven for all of the integer encoding schemes. The bound achieved in each case is the same as the formula for f with an entropy replacing each $\log i$.

We can use Corollary 2 to prove Shannon's source coding theorem (see Gallager [1968]). This theorem says that the number of bits per symbol needed to transmit the information from a discrete memoryless source can be made as close as desired to the entropy of the source. Let X be a sequence generated by a discrete memoryless source s. By grouping the symbols of X in blocks of size k and using the Move-to-Front scheme of Corollary 2 on these blocks, the average number of bits per block used is bounded by $1+H(s^k)+2\log(1+H(s^k))$, where $H(s^k)=kH(s)$ is the entropy of the block source s^k. Hence the average number of bits per symbol of X is bounded above by $1/k+H(s)+(2/k)\log(1+k\,H(s))$. As k goes to infinity this number approaches the entropy of the source, proving the theorem.

REFERENCES

Bentley, J. L. and C. A. McGeoch [1982]. "Worst-case analysis of self-organizing sequential search heuristics", *Proc. of the 20th Allerton Conference on Communication, Control, and Computing* pp. 452-461.

Bentley, J. L. and A. C. Yao [1976]. "An almost optimal algorithm for unbounded searching", *Information Processing Letters 5*, 3, August 1976, pp. 82-87.

Elias, P. [1975]. "Universal codeword sets and representation of the integers", *IEEE Transactions on Information Theory IT-21*, pp. 194-203.

Gallager, R, G. [1968]. *Information Theory and Reliable Communication*, New York: Wiley, 1968.

Gallager, R., G. [1978]. "Variations on a theme by Huffman", *IEEE Transactions on Information Theory, IT-24*, 6, November 1978, pp. 668-674.

Hardy, Littlewood, and Polya [1967]. *Inequalities*, Cambridge University Press.

Huffman, D. A. [1951]. "A method for the construction of minimum redundancy codes", *Proceedings of the IRE 40*, pp. 1098-1101.

Sleator, D. D. and R. E. Tarjan [1984]. "Amortized efficiency of list update rules", *Proceedings of the Sixteenth Annual Symposium on the Theory of Computing*, pp. 488-492.

Thompson, C. [1984]. *Private communication.*

Ziv, J. and A. Lempel [1978]. "Compression of individual sequences via variable-rate coding", *IEEE Transactions on Information Theory IT-24*, 5, September 1978, pp. 530-536.

ON THE USE OF SMOOTHSORT AND STOCHASTIC MOVE-TO-FRONT OPERATIONS FOR OPTIMAL LIST ORGANIZATION

B. JOHN OOMMEN
School of Computer Science, Carleton University,
Ottawa, Ontario, Canada. K1S 5B6

ABSTRACT

Consider a list of elements $\{R_1,\ldots,R_N\}$ in which the element R_i is accessed with an (unknown) probability s_i. If the cost of accessing R_i is proportional to i (as in sequential search) then it is advantageous if each access is accompanied by a simple reordering operation. This operation is chosen so that ultimately the list will be sorted in the descending order of the access probabilities.

We present a dynamic list organizing scheme which permits the element R_j to be moved to the front of the list on being accessed. However, as opposed to the schemes discussed in the literature, the move operation is performed with probability p_j. The latter quantity adaptively evolves in such a way that in the limit, no more move operations are performed. When this occurs we say that the scheme has converged. Although the scheme could converge to anyone of the N! possible configurations, we shall show that by suitably updating the probabilitys $\{p_j\}$ the probability of converging to the right arrangement can be made as close to unity as desired.

INTRODUCTION

Suppose we are given a set of elements $\{R_1,\ldots,R_N\}$. At every time instant one of these elements is accessed. Further, the element R_j is accessed with an unknown access probability s_j. We assume that the accesses are made independently. Whenever an element R_j is accessed, a sequential search is performed on the list. To minimize the cost of accessing, it is desirable that the records are ordered in the descending order of their access probabilities. We shall refer to a file ordered in this way as a completely organized file.

McCabe [11, pp.398-399, 12] was the first to propose a solution to this problem. His solution rendered the list dynamically self-organizing and it involved moving an element to the front of the list every time it was accessed. Using this rule, the Move-to-Front (MTF) rule, the limiting average value of the number of probes done per access has the value C_{MTF}, where,

$$C_{MTF} = 0.5 + \sum_{i,j} (s_i\ s_j)/(s_i + s_j)$$

Many other researchers [3, 6-9, 14] have also extensively studied the MTF rule and various properties of its limiting convergence characteristics are available in the literature.

McCabe [11, pp.398-399, 12] also introduced a scheme which is called the transposition rule. This rule requires that an accessed element is interchanged with its preceding element in the list, unless, of course, it is at the front of the list. Much literature is available on the transposition rule [1,2,4,11,12,13] but particularly important is the work of

Rivest [13] and Tenenbaum et al [1,15] who extensively studied this rule and suggested its generalization - the Move-k-Ahead rule and the POS(k) rule. In the former, the accessed element is moved k-positions forward to the front of the list unless it is found in the first k positions - in which case it is moved to the front. The POS(k) rule moves the accessed element to position k of the list if it is in positions k+1 through n. It transposes it with the preceding element if it is in positions 2 through k. If it is the first element it is left unchanged.

Rivest [13] showed that the limiting behaviour of the transposition rule (quantified in terms of the average number of probes) was never worse than that of the MTF rule. He conjectured that the transposition rule has lower expected cost than any other reorganization scheme. He also conjectured that the move-k-ahead rule was superior to the move-k+1-ahead rule; but as yet this is unproven. Tenenbaum and Nemes [15] proved various results for the POS(k) rule primarily involving a distribution in which $s_2 = s_3 = \ldots = s_N = (1-s_1)/(N-1)$. Their results seem to strengthen Rivest's conjecture.

All of the schemes discussed in the literature are represented by Markov chains which are ergodic. By virtue of this fact, the list can be in any one of its N! configurations - even in the limit. Thus, for example, a list which was completely organized can be thoroughly disorganized by the MTF rule by a single request for the element which is accessed most infrequently. Observe that after this unfortunate occurrence, it will take a long time for the list to be organized again - i.e., for this element to dribble its way to the tail of the list.

In this paper we propose a learning algorithm in which the elements of the list adaptively learn to find their place. The algorithm is essentially a move-to-the-front algorithm with the exception that an element R_j is moved to the front of the list with a probability p_j. This probability is systematically decreased every time the element is accessed. Ultimately on being accessed, each element tends to stay in the place where it is (as opposed to moving to the front of the list). In other words, the Markovian representation of the procedure is absorbing, as opposed to ergodic. The organization of the list gets "absorbed" into one of the N! orderings. However, we shall show that the probability of being absorbed into the optimal arrangement can be made as close to unity as desired.

Apart from the algorithm presented by us being much more accurate than all the algorithms reported in the literature, it is computationally more efficient. This is because the access of an element requires the update of exactly one probability (namely the probability p_j associated with the accessed element R_j). Further, unlike the contemporary algorithms, since the list operations are essentially stochastic, a list operation is not necessarily performed on every access. Finally, since the Markovian representations of the scheme is absorbing, the number of list operations performed asymptotically decreases to zero.

Simulation results for various distributions of $\{s_j\}$ indeed demonstrate the optimal properties of the scheme proposed.

As in the literature concerning the theory of adaptive learning, we shall use the terms algorithm, rule and scheme interchangeably.

PROBABILISTIC MOVE-TO-FRONT OPERATIONS

The concept of performing probabilistic move operations on an accessed element is now entirely new. Kan and Ross [9] suggested a probabilistic transposition scheme and showed that no advantage was obtained by rendering

the scheme probabilistic. Their scheme, however, required that the probability of performing the operation, be time invariant. As opposed to this, we shall define move operations which are essentially probabilistic, but the probabilities associated with the move operations are dynamically varied.

Let $p_i(n)$ be the probability (at time 'n') of the element R_i being moved to the front of the list on being accessed. Observe that this implies that the element, on being accessed, stays where it is with probability $(1-p_i(n))$. For an initial condition we define,

$$p_i(0) = 1, \quad i = 1,\ldots,N. \tag{1}$$

The probability $p_i(n)$ is updated every time the record R_i is accessed. The updating scheme is given by (2) below, for $0<a<1$.

$$\begin{aligned} p_i(n+1) &= ap_i(n) && \text{if } R_i \text{ is accessed} \\ &= p_i(n) && \text{if any other } R_j \neq R_i \text{ is accessed} \end{aligned} \tag{2}$$

The quantity 'a' is defined as the updating constant. We shall later discuss how this constant is chosen. Observe that the updating scheme is reminiscent of the Linear Reward-Inaction scheme studied extensively in the area of adaptive learning.

We now derive the properties of $p_i(n)$. To simplify notation, unless explicitly stated, p_i will refer to the quantity $p_i(n)$.

Theorem I

$E[p_i(n)]$ decreases monotonically with time. Further, for all $j \neq k$, $s_j > s_k$ if and only if $E[p_j(n)] < E[p_k(n)]$.

Proof: Consider the random variable $p_i(n+1)$. By virtue of (2), the latter has the following distribution:

$$\begin{aligned} p_i(n+1) &= ap_i, && \text{w. prob. } s_i \\ &= p_i && \text{w. prob. } (1-s_i) \end{aligned} \tag{3}$$

$$\begin{aligned} \text{Thus, } E[p_i(n+1)\,|\,p_i] &= ap_is_i+p_i(1-s_i) \\ &= p_i(1-(1-a)s_i) \\ &= e_ip_i, \qquad \text{where } e_i = 1-(1-a)s_i. \end{aligned} \tag{4}$$

Note that since $0<a<1$, for all i, e_i obeys $0<e_i<1$. Taking expectations again, we obtain,

$$E[p_i(n+1)] = e_iE[p_i(n)]. \tag{5}$$

The difference equation (5) subject to the initial condition (1), yields the solution,

$$E[p_i(n)] = e_i^n.$$

Clearly, $E[p_i(n)]$ is monotonically decreasing with n. Further,

$$s_j > s_k \iff e_j < e_k \iff e_j^n < e_k^n,$$

and the theorem is proved. ***

Corollary 1. For all i,

$$\lim_{n\to\infty} p_i(n) = 0 \qquad \text{w. prob. 1.}$$

The result follows since the Markov process $\{p_i(n)\}$ has only one absorbing

barrier, namely the probability 0 [10].

Theorem II

For $n>0$, Var $[p_i(n)]$ decreases monotonically with n and its limiting value is zero.

Proof: From the distribution of $p_i(n)$ given by (3), we obtain,

$$p_i^2(n+1) = a^2 p_i^2 \quad \text{w. prob. } s_i$$
$$= p_i^2 \quad \text{w. prob. } (1-s_i).$$

Thus, $E[p_i^2(n+1)|p_i] = [1-(1-a^2)s_i]p_i^2$. (6)

Taking expectations again we obtain the difference equation,

$$E[p_i^2(n+1)] = [1-(1-a^2)s_i]\, E[p_i^2(n)].$$

Since $E[p_i^2(0)] = 1$, we have,

$$E[p_i^2(n)] = [1(1-a^2)s_i]^n. \quad (7)$$

Consider now the expression for Var $[p_i(n+1)|p_i]$. Using the form of $E[p_i(n+1)|p_i]$ from Theorem I, we obtain,

$$\text{Var}[p_i(n+1)|p_i] = E[p_i^2(n+1)|p_i] - E^2[p_i(n+1)|p_i]$$
$$= [1-(1-a^2)s_i]\, p_i^2 - [1-(1-a)]^2 p_i^2$$
$$= (1=a)^2\, s_i(1-s_i)p_i^2.$$

Taking expectations again and using (7), we get

$$\text{Var}[p_i(n+1)] = (1-a)^2\, s_i(1-s_i)\, [1-(1-a^2)s_i]^n$$

which monotonically decreases with n and has a limit of zero as $n\to\infty$. ***

Remark: The fact that the limiting value of $\text{Var}[p_i(n)]$ is zero is obvious from Theorem I, since $p_i(n)$ is a nonnegative random variable whose limiting mean is zero. However, the fact that it monotonically decreases to this limiting value is not necessarily obvious, and this is the main contribution of Theorem II.

Summarizing the results of the above theorems we note that given two elements R_j and R_k, the probabilities of p_j and p_k converge w.p.1 to zero. Further, at any instant,

$$E[p_j(n)] > E[p_k(n)] \text{ if and only if } s_j < s_k.$$

Observe that this is true for all $0<a<1$. By appropriately choosing the value of the updating constant 'a', we shall now show that a stochastically stronger inequality exists - which not merely relates the expected values of p_j and p_k but the probabilities themselves.

Theorem III

For all j, k where $j \neq k$, if $s_j > s_k$, then, the quantity $\Pr[p_j(n)<p_k(n)]$ can be made as close to unity as desired.

Proof: Assume with no loss of generality that $s_j > s_k$. Let,

$$x_{j,k}(n) = p_j(n)/p_k(n).$$

Clearly, $x_{j,k}(0)=1$. Further, $x_{j,k}(n)$ can only assume non-negative values.

Consider the distribution of $x_{j,k}(n+1)$ given the values of $p_j(n)$ and $p_k(n)$. By virtue of (2),

$$x_{j,k}(n+1) = \begin{cases} (a\ p_j)/p_k & \text{w. prob. } s_j, \\ p_j/(a\ p_k) & \text{w. prob. } s_k, \\ p_j/p_k & \text{w. prob. } (1-s_j-s_k) \end{cases}$$

$$\text{Thus, } x_{j,k}(n+1) = \begin{cases} a x_{j,k}(n) & \text{w. prob. } s_j, \\ x_{j,k}(n)/a & \text{w. prob. } s_k, \\ x_{j,k}(n) & \text{w. prob. } (1-s_j-s_k). \end{cases} \tag{8}$$

Taking conditional expectations yields,

$$E[x_{j,k}(n+1)\ x_{j,k}(n)] = 1/a\ [a^2 s_j + a(1-s_j-s_k)+s_k]\ x_{j,k}(n).$$

Whence, on taking expectation again and observing that $x_{j,k}(0)=1$, we get,

$$E[x_{j,k}(n)] = (h_{j,k})^n, \text{ where } h_{j,k} = 1/a\ [a^2 s_j + a(1-s_j-s_k)+s_k]. \tag{9}$$

Let a be any real number satisfying, $s_k < a\ s_j$. Then,

$$s_k/a < s_j$$

$$\Rightarrow s_k(1-a)/a < s_j(1-a)$$

$$\Rightarrow s_k(1/a - 1) < s_j(1-a)$$

$$\Rightarrow a s_j - s_j - s_k + (s_k/a) < 0$$

$$\Rightarrow h_{j,k} < 1.$$

Thus, $h_{j,k}$ can be made strictly less than unity by appropriately choosing 'a'. From (9), this implies that $E[x_{j,k}(n)]$ can be rendered monotonically decreasing with n, and further,

$$\lim_{n\to\infty} E[x_{j,k}(n)] = 0$$

But $x_{j,k}(n)$ is always non-negative. This implies that

$$\lim_{n\to\infty} x_{j,k}(n) \to 0 \qquad \text{w. prob. } 1$$

Thus, $\lim \Pr[x_{j,k}(n) > 0] \to 0$ w. prob. 1,

and the result follows. ***

Remark: Observe that although the values of $\{s_i\}$ are unknown, the above theorem says that by making 'a' sufficiently close to unity the asymptotic value $\Pr[p_j(n) < p_k(n)]$ can be made as close to zero as desired.

A STOCHASTIC MOVE-TO-FRONT ALGORITHM

Using the results of Theorems I-III we shall now propose a stochastic move-to-front algorithm which is optimal for all distributions of access probabilities.

Initially, $p_i(0)$ is set to unity for all i. As each request is processed, the accessed element is moved to the front of the list with a probability $p_i(n)$ and the value of $p_i(n+1)$ is computed. After a sufficiently long sequence of accesses, the elements are sorted in the ascending order of the probabilities $p_i(n)$. Note that by this time the quantities $p_i(n)$ will be sufficiently small and thus the probability of getting out of this arrangement will be correspondingly small.

Rather than wait for a long period of time we propose to sort the list of elements repeatedly after every T accesses. T is called the Periodicity of Sorting, and a sequence of T consecutive accesses is called a Cycle. Initially the list is completely unsorted and so in O(N log N) time the elements can be arranged in the ascending order of $p_i(n)$. However, as the experimental results of the next section prove, after the list has been sorted a few times, the list remains "almost sorted" subsequently. This is because of the fact that although the p_i's do change subsequently, the number of move to the front operations performed diminish considerably.

Since the list remains "almost sorted", a sorting algorithm which performs well in such an environment should be used to perform the sorting. Dijkstra's Smoothsort [5] lends itself ideally to such an application. The latter is an excellent sorting algorithm which sorts a (nearly) sorted list in O(N) time and a completely unsorted list in O(N log N) time. Further, there is a smooth transition in the complexity of the algorithm as the number of unsorted elements increase. The hybrid algorithms proposed by Cook and Kim [4] could also be used to perform the sorting. Observe that the work done to sort the list is ultimately marginal - since it takes O(N) time to access the element at the tail of the list - which is the order of the time taken to sort the "almost sorted" list [5].

For the sake of completeness we now present algorithmically the list organizing scheme which uses stochastic Move-to-Front operations.

Algorithm I

Input: The updating constant 'a' and the Periodicity of Sorting T*
Output: A list which is dynamically reorganized using stochastic Move-to-Front operations.
Method: Initialize $p_i=1$ for i=1 to N
Repeat
for a sequence of T requests
move accessed element R_i to the front with probability p_i
$p_i=ap_i$
endfor
Smoothsort the list in ascending order of $\{p_j\}$
forever
End

With regard to computation, Algorithm I is more efficient than any of the algorithms known to us. This is because, as stated earlier, an access involves the modification of only one probability (which is the probability p_i associated with the accessed element R_i). Besides this, because the list operations are stochastic, a list operation is not necessarily performed on every access. Further, since the Markovian representation of the scheme is absorbing, the number of list operations performed will asymptotically decrease to zero.

Remark: (1) It can be observed that at the instances of sorting the algorithm performs exactly as an algorithm which sorts the records in the descending order of the number of times the individual records are accessed. However, in between two consecutive sorts, the algorithm presented by us does indeed manipulate the list - the manipulation being that of stochastically moving a particular element to the front of the list. In this aspect the technique differs from that of merely maintaining counters.

* It is possible to modify the algorithm so that the subsequent calls to the sorting routine are less frequent as the algorithm converges. This can be achieved by increasing T systematically every time the sorting routine is called. However, the asymptotic properties of either arrangement would be the same.

(2) The scheme presented by us has one drawback - and that is the extra memory requirement. Since the probability of moving an element to the front of the list is monotonically non-increasing, this quantity could be ultimately arbitrarily small. The technique requires that this quantity will have to be stored for every record. This could prove to be a major criticism of the technique - in spite of the fact that it is theoretically conceivable. (3) An unsolved problem is that of obtaining an optimal stochastic list organizing strategy which does not require the sorting of any index associated with the records.*

EXPERIMENTAL RESULTS

The list organizing scheme proposed in this paper was tested for files with various numbers of records and three types of distributions defined below as DIST1, DIST2, and DIST3.

DIST1: $s_i = k_1.(N-i+1)$, where, $k_1 = 1/(\sum_{i=1}^{N} i)$.

DIST2: $s_i = k_2/i$, where, $k_2 = 1/(\sum_{i=1}^{N} 1/i)$.

DIST3: $s_i = k_3/(2^i)$, where, $k_3 = 1/(\sum_{i=1}^{N} 1/2^i)$.

These distributions have been extensively studied in the literature [11]. Note that DIST2 is the familiar Zipf's law.

Ten experiments, each consisting of 4000 requests were performed on lists of various sizes. The periodicity of sorting, was 100. As a measure of the performance of the scheme, the accuracy of the scheme was computed in terms of the number of distinct pairs which were correctly organized. We refer to this measure as ACC, where,

$$ACC = \frac{\text{No. of distinct pairs correctly organized}}{\binom{N}{2}}$$

To measure the convergence characteristics and the process of stabilization of the organization of the list, we have also evaluated a quantity, K_T, which is the number of moves to the front that are performed per cycle. In other words, K_T is the number of times the list is manipulated between calls to the sorting routine.

Plots of the average values of ACC and K_T for N=10 and 40 are given in Figures I and II respectively. The distribution of the access probabilities in these figures obeyed Zipf's law. In the case when the number of records was 10, the initial average value of ACC was 0.55. After the first sort it rose to 0.93, and within the next twelve cycles the average value of ACC settled at its terminal value of 0.96. The average value of K_T was initially 29.50. After the first sort it fell to 5.10 and then, before the next 5 cycles, its value fell to less than one. This means that on the average, the list was manipulated less than once per cycle after 6 calls to the sorting routine.

Simulation results for N=10,15,25 and 40 and for all three types of distributions are given in Table I. These results clearly demonstrate the power of the scheme we have proposed.

---

* Please see postscript at the end of this paper.

N	Type of Distribution	$\hat{ACC}(0)$	$\hat{ACC}(10)$	$\hat{K}_T(0)$	$\hat{K}_T(10)$
10	DIST1	0.48	0.92	35.10	0.30
	DIST2	0.55	0.94	29.50	0.40
	DIST3	0.41	0.90	18.20	0.30
15	DIST1	0.50	0.89	43.00	0.70
	DIST2	0.50	0.90	33.80	1.00
	DIST3	0.51	0.83	21.50	0.20
25	DIST1	0.52	0.87	52.00	1.00
	DIST2	0.50	0.88	36.50	1.10
	DIST3	0.49	0.83	18.60	0.40
40	DIST1	0.52	0.87	60.80	0.90
	DIST2	0.48	0.87	37.00	1.40
	DIST3	0.46	0.87	19.70	0.40

TABLE I: Simulation Results for the Stochastic Move-to-Front List Organizing Scheme

Notation:

$\hat{N}$: No. of elements in the list.

$\hat{ACC}(k)$: Average value of the percentage of the number of pairs which are correctly organized after the kth sort.

$\hat{K}_T(k)$: Average value of the number of move to the front operations performed in the kth cycle.

DIST1: $s_i = k_1.(N-1+1)$, where, $k_1 = 1/(\sum_{i=1}^{N} i)$.

DIST2: $s_i = k_2/i$, where, $k_2 = 1/(\sum_{i=1}^{N} 1/i)$.

DIST3: $s_i = k_3/(2^i)$, where, $k_3 = 1/(\sum_{i=1}^{N} 1/2^i)$.

CONCLUSIONS

We have considered a list of elements $\{R_1,\ldots,R_N\}$ in which the element R_i is accessed with a probability s_i. The latter is unknown apriori. A list organizing scheme has been proposed. The scheme performs a move to the front operation on the accessed element R_j with a probability p_j which is systematically decreased. Ultimately, the list gets absorbed into one of the N! possible arrangements. We have shown that the asymptotic probability of converging to the optimal arrangement can be made as close to unity as desired.

ACKNOWLEDGEMENTS

I am greatly indebted to my colleague Nicola Santoro who introduced me to the problem and to the literature in the field. I am grateful to him for his encouragement and comments during the course of the study. I would also like to acknowledge my gratitude to Jorg Sack and Thomas Strothotte who informed me of Dijkstra's Smoothsort algorithm. Finally, I would like to thank my colleague Michael Atkinson for his critical reading of the

manuscript and his comments which rendered the paper more readable. This research work was partially supported by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] Arnow, D.M. and Tenebaum, A.M., "An Investigation of the Move-Ahead-K Rules", Congressus Numerantium, Proc. of the Thirteenth Southeastern Conference on Combinatorics, Graph Theory and Computing, Florida, February 1982, pp.47-65.

[2] Bitner, J.R., "Heuristics That Dynamically Organize Data Structures", SIAM J. Comput., Vol.8, 1979, pp.82-110.

[3] Burville, P.J. and Kingman, J.F.C., "On A Model For Storage and Search", J. Appl. Probability, Vol.10, 1973, pp.697-701.

[4] Cook, C.R., and Kim, D.J., "Best Sorting Algorithm For Nearly Sorted Lists", Comm. ACM, Vol.23, 1980, pp.620-624.

[5] Dijkstra, E.W., "Smoothsort, An Algorithm For Sorting In SITU", Science of Computer Programming, 1982, pp.223-233.

[6] Gonnet, G.H., Munro, J.I. and Suwanda, H., "Exegesis of Self Organizing Linear Search", SIAM J. Comput., Vol.10, 1981, pp.613-637.

[7] Hendricks, W.J., "The Stationary Distribution Of An Interesting Markov Chain", J. App. Probability, Vol.9, 1972, pp.231-233.

[8] Hendricks, W.J., "An Extension Of A Theorem Concerning An Interesting Markov Chain", J. App. Probability, Vol.10, 1973, pp.231-233.

[9] Kan, Y.C. and Ross, S.M., "Optimal List Order Under Partial Memory Constraints", J. App. Probability, Vol.17, 1980, 1004-1015.

[10] Karlin, S. and Taylor, H.M., "A First Course in Stochastic Processes", Academic Press, 1975.

[11] Knuth, D.E., "The Art of Computer Programming, Vol.3, Sorting and Searching", Addison-Wesley, Reading, Mass., 1973.

[12] McCabe, J., "On Serial Files With Relocatable Records", Operations Research, Vol.12, 1965, pp.609-618.

[13] Rivest, R.L., "On Self-Organizing Sequential Search Heuristics", Comm. ACM, Vol.19, 1976, pp.63-67.

[14] Sleator, D. and Tarjan, R., "Amortized Efficiency of List Update Rules", Proc. of the Sixteenth Annual ACM Symposium On Theory of Computing, April 1984, pp.488-492.

[15] Tenenbaum, A.M. and Nemes, R.M., "Two Spectra of Self-Organizing Sequential Search Algorithms", SIAM J. Comput., Vol.11, 1982, pp.557-566.

POSTSCRIPT: After the preparation of this manuscript it was brought to our attention that the scheme proposed here is potentially worse than the scheme which only uses counters, although both of them perform identically at sorting instances. This is because, although the stochastic move-to-front operations performed in between the sorting instances would ultimately diminish, as time proceeds, the operations tend to be performed on the less frequently accessessed elements of the list. This renders the list more disorganized than it would have been if it was not manipulated using the stochastic move-to-front operations in between the sorting instances. The actual quantification of how this scheme performs in between sorting instances is yet unsolved.

Amusingly enough, stochastic move-to-back operations do not have this drawback.

We are extremely grateful to the anomymous reviewers who brought this to our attention.

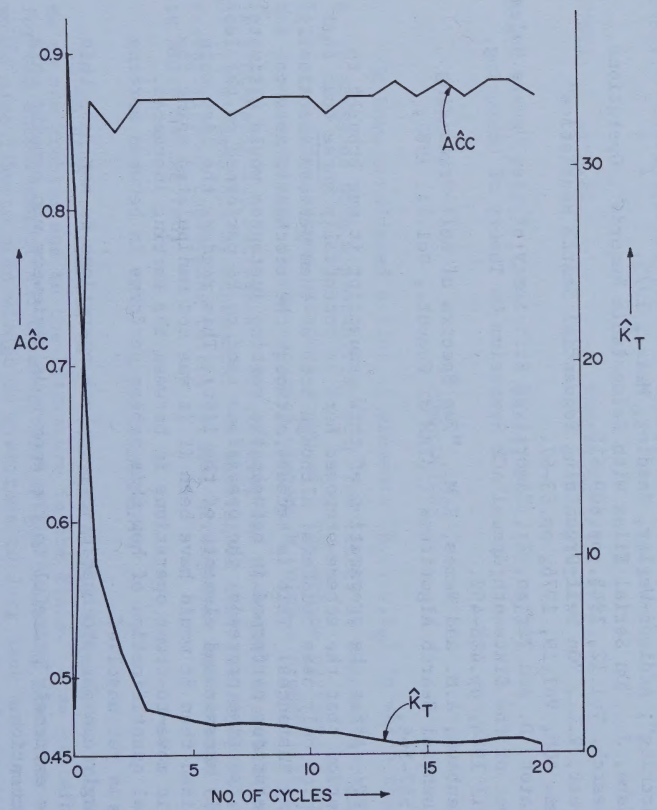


Figure II : Variation of $\hat{ACC}$ and $\hat{K}_T$ with time for N = 40.
The parameters of the scheme are T=100 and a=0.9.
The Access Probabilities obey Zipf's Law.

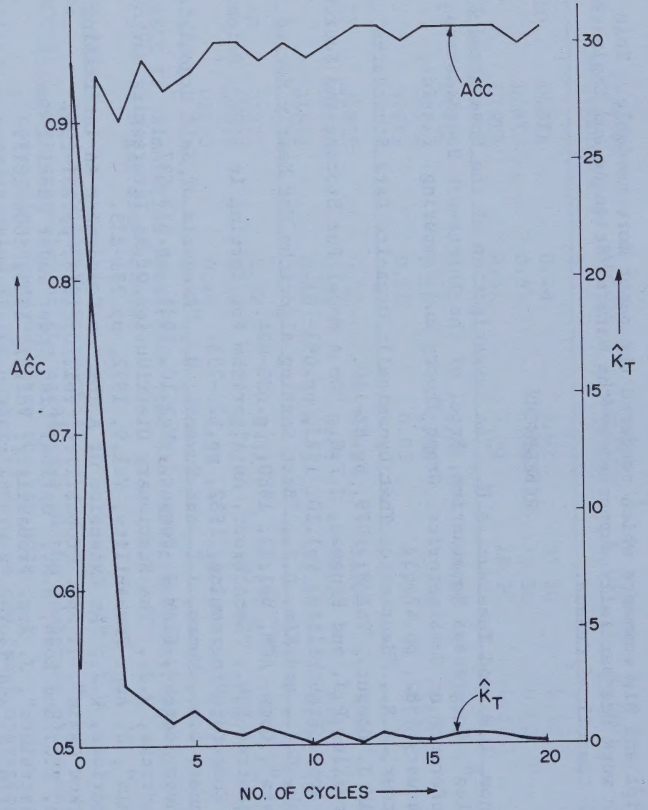


Figure I : Variation of $\hat{ACC}$ and $\hat{K}_T$ with time for N = 10.
The parameters of the scheme are T=100 and a=0.9.
The Access Probabilities obey Zipf's Law.

MANAGING STORAGE FOR EXTENDIBLE ARRAYS OF VARYING DIMENSIONS

Ekow J. Otoo
School of Computer Science, Carleton University, Ottawa, Canada.

Abstract

We introduce a method for managing storage of dense rectangular arrays in consecutive memory locations such that the number of dimensions as well as the index range of each dimension can be extendible. The method realizes an n-element array of any shape in $\Theta(n)$ storage locations with a computed access function of complexity $O(d^2)$, where n and d are respectively, the size and dimensionality of the array at any instant.

This presents a solution to a more difficult variant of an open question of Rosenberg that asks whether or not a realization exists which favours non-trivial infinite family of shapes such that an array of n elements is spread over at most $C * n$ locations for some integer C. We show that $C \leq 3$ for any d.

1. Introduction

Computed access techniques for mapping elements of dense rectangular arrays of bounded size and fixed number of dimensions in consecutive storage locations have the virtue of fast element access and efficient storage utilization. Unfortunately they lack the flexibility of admitting extensions in the size of the dimensions except for one. Furthermore, the array must be defined to be of a fixed number of dimensions that cannot increase during the life time of the array. There is no known strategy that uses computed access method for allocating and retrieving elements of arrays that not only allows extension of the range of subscript values but allows the number of dimensions to grow.

Let N be the set of non-negative integer. A declaration of an array is of the form $A[l_1 : u_1, l_2 : u_2, \ldots, l_d : u_d]$ where $l_j\ u_j \in N$, $j = 1, 2, \ldots, d$. An element of the array is referred to as $A\langle i_1, i_2, \ldots, i_d\rangle$, for $l_j \leq i_j \leq u_j$. An array is realized in storage as a set of consecutive memory locations $M[0 : n-1]$, where $n = (\prod_{j=1}^{d} (u_j - l_j + 1))$. Suppose an element $A\langle i_1, i_2, \ldots, i_d\rangle$ is stored in location $M\langle q\rangle$ then the index q is given by a mapping function $\Re : N^d \rightarrow N$, such that $q = \Re(\langle i_1, i_2, \ldots, i_d\rangle)$. The mapping function $\Re$ maps the d-tuple indices one-to-one onto the n consecutive locations. This method of realizing arrays is referred to as a computed access method (see [12]). Typical computed access methods for bounded rectangular arrays, are exemplified by the conventional row-major (or lexicographic) order and the column-major order of element allocations.

A drawback in such conventional techniques, such as the row-major order or column-major order, is that they allow the expansion of the array only along one dimension. Rosenberg [13] refers to arrays realized in this way as "prism arrays".

Consider first, the case when the array is of fixed number of dimensions. One can envisage a scenerio in which the upper bounds of the range of index values of each dimension may be progressively increased on demand. Assuming a d-dimensional array is initially declared as $A[l_1 : u_1, \ldots, l_j : u_j, \ldots, l_d : u_d]$. Then at some point in time, the dimension j is extended so that the new array becomes $A[l_1 : u_1, \ldots, l_j : u_j + 1, \ldots, l_d : u_d]$. The effect of the expansion

is to admit a (d-1)-dimensional subarray of size $\prod_{\substack{r=1 \\ r \neq j}}^{d} (u_r - l_r + 1)$ elements. A computed access method for realizing extendible arrays requires that the new set of elements be allocated in conservative locations following the block of locations occupied by the previous array and that the mapping function correctly addresses every element of the array without redefinition or storage reorganization.

1.1 Related Works.

Pioneering works in techniques for realizing extendible arrays of fixed dimensionality are presented in a series of papers by Rosenberg [12, 13], Stockmeyer [16] and also jointly by Rosenberg and Stockmeyer [14]. They examined extendible arrays in the light of their work on data graphs and addressing schemes. Their work investigated extendible arrays using four criteria for measuring the quality of a realization, viz., simplicity of arbitrary element access, simplicity of traversal along a path, simplicity of extension and efficiency of storage utilization.

It is shown in [13] that the lower bound on the storage requirement, for an extendible array of n elements, for a storage mapping with unbridled extendibility is $O(n\,(lg\,n)^{d-1})$. However, by restricting the pattern of expansion of the array and allowing only arrays that conform to the restriction, the storage requirement is improved to $O(n)$. Two methods of storing n-element extendible arrays with restricted pattern of growth in $O(n)$ locations are demonstrated by the diagonal and cubical shell index realizations [12,13].

Hansson and Tärnlund [2] present another approach for realizing arrays having a prescribed pattern of growth. The technique called the "Zig-zag procedure" achieves 100% storage efficiency since the array is constrained to be always cubical at each expansion step.

Similar storage efficient realizations for arrays with restricted pattern of growth have been demonstrated by our earlier works on storing extendible arrays whose range of subscript values can be extended either linearly or exponentially (see [6, 7,8]). Ouksel and Scheuermann [11] have presented a method of representing cubical arrays having linear expansions, such that an n element array is represented in n locations. The mapping function however, is computable in $O(log\,n)$ which is not independent of the size of the array.

In [13] the question was asked regarding the lower bound on the storage utilization of extendible array realization that admit infinitely may shapes. In particular, the author asks whether or not a realization exists which favours, non-trivially, infinite family of shapes such that an array of n element is spread over $C*n$ locations. The paper further reported a suggestion by Shmuel Winograd who asked whether added information might allow one to overcome the $O(n\,lg\,n)$ lower bound. We have shown in an earlier paper [10], in introducing the "Index-Array" technique for extendible array, that the answer is affirmative.

The Index-Array technique allocates a d-dimensional array of n elements using $\Omega(n)$ locations in the best case and $O(n*d^2)$ locations in the worst case. The complexity of the mapping function in the Index-Array scheme is $\Theta(d)$. Essentially the Index-Array scheme computes the address of an element $A\langle i_1, i_2, \ldots, i_d\rangle$ using a row-major order function of the form

$$\Re(\langle i_1, i_2, \ldots, i_d\rangle) = s_0 + c_1\,i_1 + c_2\,i_2 + \ldots + c_d\,i_d\,.$$

for any d-tuple index. The value of s_0 is the starting address of some block of elements in which $A\langle i_1, i_2, \ldots, i_d\rangle$ belongs. The values $s_o, c_1, c_2, \ldots, c_d$ are information stored in locations given by a 3-dimensional array called the Index-Array. This added information enables the Index-Array scheme to overcome the $O(n\,(lg\,n)^{d-1})$ lower bound as rightly suggested by Winograd.

Linked allocation techniques, such as the use of vector of pointers, linked list, orthogonal list (see [4,15]) and even hashing techniques [14] may be used in array representations that admit extendibility. What then makes computed access methods attractive ?

The answer is in the advantages that may be derived and the application of the realization functions to other data organization schemes. Two advantages of computed access realization

are stated in [13], viz., i) such array realizations afford one both easy probing of the array, which are lacking in linked schemes, and ii) they allow easy traversal along, say, rows and columns of the array, which is not present in hashing schemes. Issues such as proximity of array elements and ease of traversal are discussed in [1] and [14].

A third reason for the quest of computed access techniques is that the mapping functions that are derived have applications in multidimensional dynamic hashing schemes for physical data organization. This is evident in the multidimensional storage schemes presented in [6, 7, 8, 9, 11]. The use of the mapping functions for extendible arrays to databases constitutes the primary motivation for our research.

The earlier works on extendible arrays have only considered the case where the number of dimensions is kept fixed, but the range of subscript values gradually increases. We term expansions of this type as "longitudinal" . An array that has the propensity to increase the number of dimension is said to admit "lateral" expansion. We present in this paper a method for managing storage for extendible arrays that allow both longitudinal and lateral expansions.

1.2 Summary of Main Results.

For simplicity the elements of the array will be considered to be integers, and a unit of storage will be taken to be number of bytes required for an integer. We propose a method , called the $\alpha\beta\gamma$-Index-Array technique, for realizing extendible arrays of varying dimensionality. Let d and n denote the dimensionality and number of elements of an array representation using this scheme. Then the method has the property that

i) The complexity of computing an arbitrary element address is $\Theta(d^2)$.

ii) The storage requirement in the best case is $(n+d^2n^{1/d}+d)$, uints of storage locations when the array is cubical.

iii) The storage requirement in the worst case is $(n+n*d^2/2^{d-1}+d)$ units of storage locations.

If an extendible array has a predefined number of dimensions, d, the $\alpha\beta\gamma$-Index-Array scheme is conservative of storage in comparison with other representation methods that tolerate limited extension. For instance using vector of pointer technique, the corresponding storage requirement for cubical arrays and that array shape for which the $\alpha\beta\gamma$-Index-Array gives the worst case, are respectively $(n+(n-n^{1/d})/(n^{1/d}-1))$ and $(n(3-1/2^{d-1}))$.

2. Some Definitions and Notations.

We present in this section some formal concepts and a representation independent denotations of an array relevant to our discussions. Let N denote the set of non-negative integers. For $d \in N-\{0\}$ and $u_j \in N$, j = 1, 2,..., d, denote the set $\{0, 1, \ldots, u_j\}$ by m_j. Let $n = \prod_{j=1}^{d} (u_j+1)$. A bounded rectangular set N_n^d, of d-tuples is defined by the Cartesian product $m_1 \times \ldots \times m_d$. An element in N_n^d is specified by an integer d-tuple $\langle i_1, \ldots, i_d \rangle$, for $0 \leq i_j \leq u_j$, $j = 1, 2, \ldots, d$, The set of integer d-tuples in N_n^d may be perceived as the the set of all lattice points in the region of d-space defined by $m_1 \times \ldots \times m_d$. The infinite expandability of the array is induced by allowing each set m_j, to expand indefinitely.

Definition 2.1. *Given N, the set of non-negative integers, $d, u_j \in N$, for $j = 1, \ldots, d$, and $m_j = \{0, 1, \ldots, u_j\}$. A d-dimensional array $A[0 : u_1, \ldots, 0 : u_d]$, is an association between integer d-tuples and elements of a set E, such that for each d-tuple in the set $m_1 \times \ldots \times m_d$, there corresponds an element of E.*

An element of a d-dimensional array is denoted as $A\langle i_1, \ldots, i_d \rangle$, and consists of that element of E associated with $\langle i_1, \ldots, i_d \rangle$ called the d-tuple index. Each individual component i_j of the d-tuple index is referred to as a subscript.

Let $M[0 : n] = \{M\langle 0 \rangle, M\langle 1 \rangle, \ldots, M\langle n \rangle\}$ be a set of consecutive storage locations with the base location being $M\langle 0 \rangle$. An allocation function $\Re : N^d \rightarrow N$, is a mapping that assigns

to each element $A\langle i_1,\ldots,\rangle$, a distinct location $M\langle q\rangle$ where q is derived from the array index $\langle i_1,\ldots,i_d\rangle$. $\Re$ is said to realize A.

Definition 2.2. *A realization of the array* $A[0 : u_1,\ldots,0 : u_d]$, *in* $M[0 : n-1]$ *where* $n = \prod_{j=1}^{d}(u_j+1)$ *is a mapping* $\Re : N^d \rightarrow N$, *of elements of A, one-to-one onto the addresses* $\{0, 1,\ldots,n-1\}$, *such that* $\Re(\langle 0,\ldots,0\rangle) = 0$.

Longitudinal extensions in a d-dimensional array may be percieved as sequence of state transitions in which the size of each dimension is expanded by adjoining a (d-1)-dimensional block. For all purposes the allocation of elements in any adjoined block will be taken to be in lexicographic order.

Definition 2.3. *A d-dimensional extendible array is a d-dimensional array* $A_\alpha[0 : u_1, \ldots, 0 : u_d]$ *in which the upper bounds of the subscripts of the various dimensions may increase in some arbitrary order.*

Definition 2.4. *An extendible array in which the dimensionality d, is varying and takes increasing values of positive integers, is referred to as as an extendible array of varying dimensions.*

An extendible array of varying dimensions is denoted by $A^*_\alpha[0 : u_1,\ldots,0 : u_d,\ldots]$ where the asterisk ('*'), reflects the fact that the dimensionality is varying and the subscript α, reflects the arbitrary nature of the expansion process. In practice, however, some specified criteria will normally dictate the choice of the dimensions to be extended.

Definition 2.5. *A realization of an extendible array of varying dimensions* $A^*_\alpha[0 : u_1,\ldots,0 : u_d,\ldots]$, *is a function* $\Re^*_\alpha : N^d \rightarrow N$ *that maps elements of* A^*_α *one-to-one onto consecutive storage locations* $M[0 : n-1]$ *such that* $\Re^*_\alpha(\langle 0,\ldots,0\rangle) = 0$, *where* $n = \prod_{j=1}^{d}(u_j+1)$, *varies in consonance with changing values of d and* u_j*'s.*

The reader may examine examples of extendible arrays of constant and variable dimensionalities shown in the Figures 3.1a and 3.2a.

The qualities of a realization will be evaluated using primarily the following three criteria.

1. The complexity of accessing an arbitrary element. For an array of dimensionality d, size n, and a mapping function $\Re$, this measure will be denoted by $\tau(d, n; \Re)$.
2. The worst case storage utilization. This will be denoted by $\sigma(d, n; \Re)$
3. The best case storage requirement where necessary. This will be denoted by $\tilde{\sigma}\ (d, n; \Re)$.

We shall employ the order notations (O- , Θ- and Ω- notations), in the sense defined in the literature.

3. Extendible Arrays of Arbitrary Expansions.

3.1 Fixed Dimensional Arrays.

To appreciate the storage allocation technique for arrays of varying dimensionality that can be arbitrarily extended, let us first briefly outline the Index-Array scheme for extendible arrays where the number of dimensions is constant. The method is fully discussed in [10].

Let $A_\alpha[0 : u_1,\ldots,0 : u_d]$ be an extendible array. The Index-Array scheme stores A_α, called the principal array in a region of storage $M[0 : n-1]$, and maintains another 3-dimensional array $I_\alpha[1 : d, 1 : d, 0 : U_x]$, where $U_x = \max_{1\leq j\leq d}(u_j)$, called the "index array". The index array I_α, is extendible only in the last dimension and as such may be realized, using conventional column-major allocation function, in some region of storage independent of M. The use of the function of the index array is to retain relevant information at each expansion step. This information is used in the computation of the address of an element $A\langle i_1,\ldots,i_d\rangle$, in M.

Consider the elements of the array A_α to be allocated in locations $M\langle 0\rangle, M\langle 1\rangle, \ldots, M\langle n-1\rangle$, and suppose the array is extended by increasing u_j to $u_j + 1$. The array is now denoted as $A_\alpha[0 : u_1, \ldots, 0 : u_j + 1, \ldots, 0 : u_d]$. The effect is that a block of subarray element must be allocated in locations $M\langle n\rangle$, $M\langle n+1\rangle$ through $M\langle n' - 1\rangle$ where $n' = (u_j + 2) * \prod_{\substack{r=1 \\ r\neq j}}^{d} (u_r + 1)$. Since ordering of the elements in any adjoined block is in column-major order, the address q of the location $M\langle q\rangle$ to which an element $A\langle i_1, \ldots, i_d\rangle$ is assigned is given by

$$q = n + \sum_{\substack{k=1 \\ k\neq j}}^{d} c_k\, i_k, \quad \text{where} \quad c_k = \prod_{\substack{r=1 \\ r\neq j}}^{k-1} (u_r + 1); \tag{3.1}$$

Let n be the block header address of the adjoined subarray. This is the address of the element $A\langle 0, \ldots, u_j + 1, \ldots, 0\rangle$. The values c_k for $k = 1, \ldots, j-1, j+1, \ldots, d$, in the equation (3.1), are the coefficients with which the respective subscripts must be multiplied to address an element in this block. As a consequence of equation (3.1), we need to retain the values of the block header address and the required coefficients, whenever the array is extended. These are retained in the index-array I_α.

Let $I_\alpha\langle j_1, j_2, x\rangle$ denote an element of the index array $I_\alpha[1 : d, 1 : d, 0 : U_x]$. A mapping function for realizing I_α, denoted by $\Phi_\alpha : N^3 \rightarrow N$, is defined as

$$\Phi_\alpha(j_1, j_2, x) = (j_1 - 1) + (j_2 - 1) * d + x * d^2; \tag{3.2}$$

During the expansion of the array $A_\alpha[0 : u_1, \ldots, 0 : u_j, \ldots, 0 : u_d]$ to $A_\alpha[0 : u_1, \ldots, 0 : u_j + 1, \ldots, 0 : u_d]$, the starting block header address and the relevant coefficients are entered in the index array as follows. If $(u_j + 1) > U_x$, we set $U_x \leftarrow (u_j + 1)$ and extend the storage locations of I_α by d^2 locations. The block header address is entered into $I_\alpha\langle j, j, u_j + 1\rangle$, and the coefficient $c_k = \prod_{\substack{r=1 \\ r\neq j}}^{k-1} (u_r + 1)$, is entered in $I_\alpha\langle j, k, u_j + 1\rangle$ for $k = 1, \ldots, j-1, j+1, \ldots, d$.

To determine the address of an element $A_\alpha\langle i_1, \ldots, i_d\rangle$, the block in which the element belongs must first be identified. This is straight forward since an element will belong to the lattest of the adjoined blocks implied by the subscript values $i_1, i_2, \ldots, i_d$, i.e., the implied block with the maximum block header address. By comparing the index array entries $I_\alpha\langle j, j, i_j\rangle$, for $j = 1, \ldots, d$, that value of j for which the block header address is maximum is easily determined. The rest of the address computation follows immediately from equation (3.1). Let the function to compute the address of an elements be π_α. Then this is defined algorithmically as

Function $\pi_\alpha(\langle i_1, \ldots, i_d\rangle)$

begin

Determine j such that $I_\alpha\langle j, j, i_j\rangle = \max(I_\alpha\langle 1, 1, i_1\rangle, \ldots, I_\alpha\langle d, d, i_d\rangle)$;

$$\pi_\alpha \leftarrow I_\alpha\langle j, j, i_j\rangle + \sum_{\substack{k=1 \\ k\neq j}}^{d} (I_\alpha\langle j, k, i_j\rangle * i_k)\ ;$$

end.

We characterize the state of an extendible array by a (d+1)-tuple formed from the number of dimensions and the upper bounds of each dimension, i.e., $(d, u_1, \ldots, u_d)$. To illustrate the addressing method of the Index-Array scheme, consider the Figures 3.1a, and 3.1b which show a 2-dimensional principal array, and the corresponding index array. The Figure 3.1a depicts the storage layout for the following sequence of state transitions : (2; 0, 0), (2; 1, 0), (2; 1, 1), (2: 1, 2), (2; 2, 2), (2; 3, 2), (2; 3, 3). At the time of the last expansion, the upper bound of the

dimension i_2 is extended from 2 to 3. The block of elements allocated are $A_\alpha\langle 0,3\rangle$, $A_\alpha\langle 1,3\rangle$, ..., $A_\alpha\langle 3,3\rangle$. These have the respective addresses 12, 13, ..., 15. The starting block address, which in this case is 12, is stored in $I_\alpha\langle 2,2,3\rangle$ and the coefficient of the subscript i_1, which in this case equals 1, is stored in $I_\alpha\langle 2,1,3\rangle$.

Proposition 3.1. *An extendible array of fixed number of dimensions and arbitrary expansion is realizable by the Index-Array scheme, in which the mapping function $\Re_\alpha = (\pi_\alpha, \Phi_\alpha)$ is defined by two functions Φ_α and π_α, where Φ_α addresses elements of the index array and π_α addresses elements of the principal array.*

The quality of the Index-Array scheme is summarized by the Theorems 3.2 and 3.3.

Theorem 3.1. *The complexity of computing the address of an arbitrary element in the Index-Array scheme is $\tau(d,n;\Re_\alpha) = \Theta(d)$.*

Proof.
The complexity of the $\Re_\alpha$ computation will be measured by the number of elementary operations such as addition, multiplication, comparisons etc. To access an element in the index array, the evaluation of π_α takes a constant time. Computing $\Re_\alpha$ cost $d-1$ comparisons, d additions and $d-1$ multiplications, hence $\tau(d,n;\Re_\alpha) = \Theta(d)$. ∎

Theorem 3.2. *The Index-Array scheme spreads a d-dimensional extendible array of n elements in $\sigma(d,n;\Re_\alpha) = O(n\,d^2)$ locations in the worst case and in $\tilde{\sigma}(d,n:\Re_\alpha) = \Omega(n)$ locations in the best case.*

Proof.
Consider an array which is predefined as d-dimensional. If circumstances dictate that expansions occur on one dimension always, then the size of the longest dimension is n when the array has n elements. The size of the index array then is $n * d^2$. This is the maximum size that the index array can get for an n-element array. The worst case storage utilization then is

$$\begin{aligned}\sigma(d,n;\Re_\alpha) &= n + n * d^2,\\ &= O(n\,d^2).\end{aligned}$$

The minimum size of the index array occurs when the principal array is hypercubical, i.e., each size is $n^{1/d}$. The index array in this case is of size $n^{1/d} * d^2$. The best case storage requirement is

$$\begin{aligned}\tilde{\sigma}(d,n;\Re_\alpha) &= n + n^{1/d} * d^2,\\ &= \Omega(n).\end{aligned}$$ ∎

3.2 Arrays of Variable Dimensions.

Let the dimensions of the index array be labelled as j_1, j_2 and X in that order. Using the concept of the Index-Array scheme of the preceding section, the case where the principal array can expand in the number of dimensions is easily appreciated once we understand how the index array I_α can be made extendible in all j_1, j_2 and X dimensions.

Consider the 2-dimensional extendible array of Figure 3.1a and its associated index array of Figure 3.1b. Suppose the principal array, which we denote as $A^*_\alpha[0:3,0:3]$, is expanded by increasing the number of dimensions to 3. The new array, shown in Figure 3.2a, is now denoted as $A^*_\alpha[0:3,0:3,0:1]$. This expansion is treated as having occured in two steps. First is the expression of intent to treat the array as a 3-dimensional array, followed by an extension of the subscript range of the third dimension. The effect is to add a 2-dimensional subarray of 16 elements on the third dimension as shown in Figure 3.2a. The added elements in row-major order are $A^*_\alpha\langle 0,0,1\rangle$, $A^*_\alpha\langle 1,0,1\rangle$, $A^*_\alpha\langle 2,0,1\rangle$, ..., $A^*_\alpha\langle 3,3,1\rangle$, and these occupy the addresses 16 through 31.

To store the header address and the required coefficients for addressing elements in the adjoined block, the original index array, denoted now as $I^*_\alpha[1:2,1:2,0:3]$ must be expanded

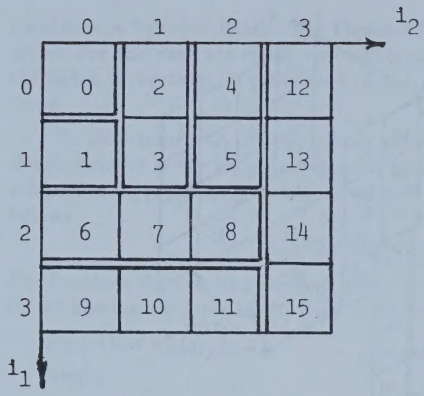


Figure 3.1a : A schematic storage layout of a 2-dimensional extend--ible array $A_\alpha[0:3, 0:3]$.

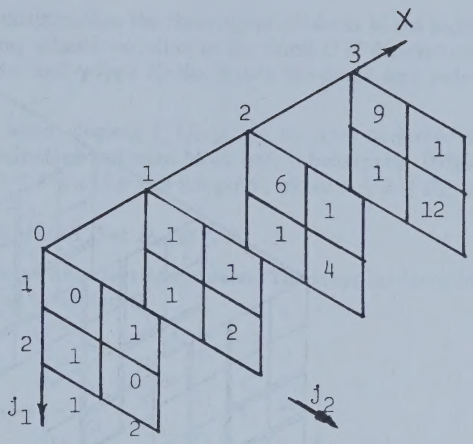


Figure 3.1b : The corresponding index array of Figure 3.1a showing the stored entries.

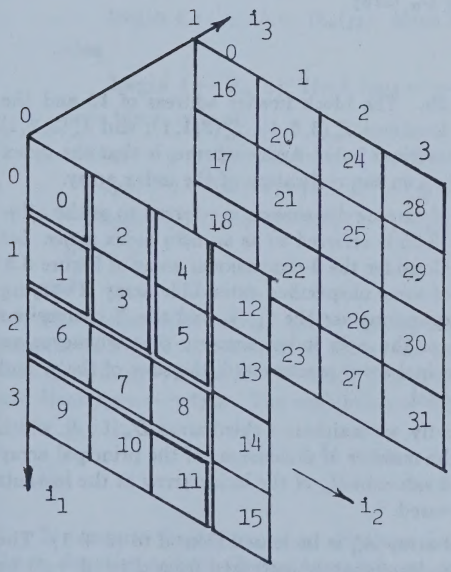


Figure 3.2a : Storage layout of a array of Figure 3.1a extended to 3 dimensions.

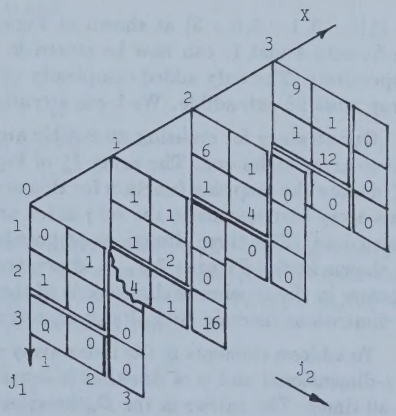


Figure 3.2b : The corresponding $\alpha\beta\gamma$ index array of Figure 3.2a showing the stored entries.

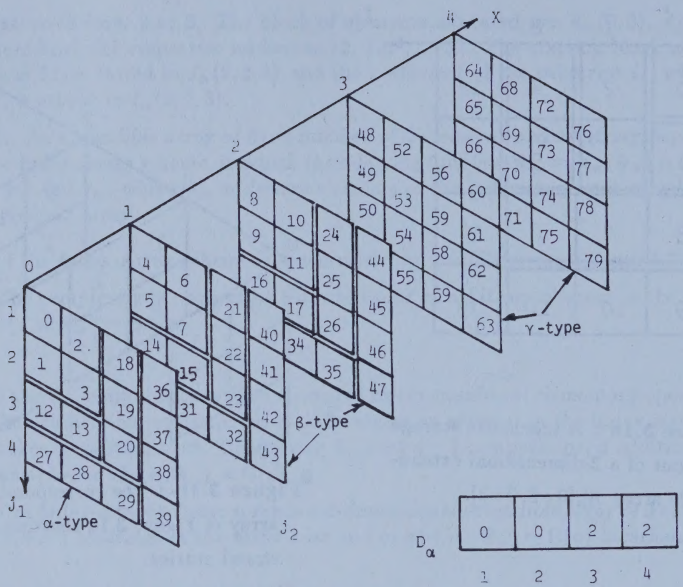


Figure 3.3 : Storage layout of an $\alpha\beta\gamma$ index array $I^*_\alpha[1:4, 1:4, 0:4]$ and its corresponding D_α array showing the stored entries.

to $I^*_\alpha[1:3, 1:3, 0:3]$ as shown in Figure 3.2b. The block header address of 16 and the coefficients 4 and 1, can now be stored in the locations $I^*_\alpha\langle 3,3,1\rangle$, $I^*_\alpha\langle 3,1,1\rangle$, and $I^*_\alpha\langle 3,2,1\rangle$ respectively. The only added complexity to the original Index-Array scheme, is that the index array must be extendible. We focus attention then on the realization of the index array.

Our strategy for realizing extendible arrays of varying dimensions is referred to as the $\alpha\beta\gamma$-Index-Array technique. The array I^*_α of Figure 3.2b is referred to as an $\alpha\beta\gamma$ index array. Let Φ^*_α denote the mapping function for this array. Consider the 3-dimensional array of Figure 3.3. This array corresponds to the $\alpha\beta\gamma$ index array of some unspecified extendible array of varying dimensions. The three directions of possible expansions are the j_1, j_2 and the X dimensions as shown in the Figure 3.3. A characteristic of the $\alpha\beta\gamma$ index array is that whenever an increase in the number of dimensions of the principal array is expressed, the sizes of the j_1 and j_2 dimensions increase such that both remain equal.

To address elements in the index array correctly, we maintain a third array $D_\alpha[1:d]$, which is 1-dimensional and is of size that is equal to the number of dimensions of the principal array at all times. The entries in the D_α array are the values of U_x of the index array at the instants the dimensionality of the principal array is increased.

Suppose the dimensionality of the principal array A^*_α is increased from d to (d + 1). The j_1 and j_2 dimensions of the $\alpha\beta\gamma$ index array are immediately increased from d to (d + 1) by adjoining a subarray of $d * U_x$ elements along the j_1 dimension followed by the adjunction of a subarray of $(d+1) * U_x$ elements along the j_2 dimension. The subarray blocks adjoined on the j_1 and j_2 dimensions are referred to as the " α-type" and "β-type" blocks respectively. The entries of the adjoined blocks of elements are all intialized to zeroes. Between any two consecutive instances of increasing the dimensionality of the principal array, extensions of the range of subscript values may occur. Such extensions may increase the size of the U_x dimension of the index array. The block of index array elements added to extend the size U_x, is said to

constitute a "γ-type" block. The Figure 3.3 distinguishes the three types of blocks of the index array. For the case where the extendible array admits variation in the number of dimensions, the index array then, is comprised of $\alpha-$, $\beta-$ and γ-type blocks, hence the name $\alpha\beta\gamma$ index array.

To determine the address of an index array element $I^*_\alpha\langle j_1, j_2, x\rangle$, we need to know the dimensionality at the time that element was admitted and what block type it belongs to. Define a function $\Psi_\alpha(x)$ that associates a value δ, $1 \leq \delta \leq d$, for any integer x, where $0 \leq x \leq U_x$, as follows.

$$\Psi_\alpha(x) = \text{maximum } d \text{ such that } D_\alpha(d) \leq x.$$

The function Ψ_α can be perceived as an inverse of a subscript function. The mapping function for addressing elements of the $\alpha\beta\gamma$ index array is defined as

Function $\Phi^*_\alpha(j_1, j_2, x)$;
begin
  if $j_1 > j_2$ **then**
    if $D_\alpha\langle j_1\rangle \geq x$ **then**
      begin $\delta \leftarrow j_1$; $h \leftarrow D_\alpha\langle j_1\rangle$; *block_type* $\leftarrow$ α-*type*; **end**;
    else
      begin $\delta \leftarrow \Psi_\alpha(x)$; *block_type* $\leftarrow$ γ-*type*; **end**
  else
    if $D_\alpha\langle j_2\rangle \geq x$ **then**
      begin $\delta \leftarrow j_2$; $h \leftarrow D_\alpha\langle j_2\rangle$; *block_type* $\leftarrow$ β-*type* ; **end**
  else
      begin $\delta \leftarrow \Psi_\alpha(x)$; *block_type* $\leftarrow$ γ-*type*; **end**
  case block_type of
    γ-*type* : $q \leftarrow (j_1 - 1) + (j_2 - 1) * \delta + x * \delta^2$;
    α-*type* : $q \leftarrow (h + 1) * (\delta - 1)^2 + (j_2 - 1) + x * (\delta - 1)$;
    β-*type* : $q \leftarrow (h + 1) * \delta * (\delta - 1) + (j_1 - 1) + x * \delta$;
  $\Phi^*_\alpha \leftarrow q$;
end ;

The computation carried out by the Φ^*_α function is illustrated with the following example. Suppose in Figure 3.3 we wish to access the location of the element $I^*_\alpha\langle 3,3,3\rangle$. We have then that $j_1 = 3$, $j_2 = 3$ and $x = 3$. Since $j_1 \leq j_2$ and $D_\alpha\langle j_2\rangle < x$, we set $\delta \leftarrow \Psi_\alpha(x) = \Psi_\alpha(3) = 4$, and *block_type* $\leftarrow$ γ-type. The evaluation of $\Phi^*_\alpha(3,3,3)$ then is given by

$$\begin{aligned}\Phi^*_\alpha(3,3,3) &= (j_1 - 1) + (j_2 - 1) * \delta + x * \delta^2; \\ &= 2 + 2 * 4 + 3 * 4^2 = 58.\end{aligned}$$

To compute the address of an arbitrary element in the $\alpha\beta\gamma$-Index-Array scheme, we apply the π_α function as in the Index-Array scheme, except that now elements in the $\alpha\beta\gamma$ index array are addressed using the function Φ^*_α.

Proposition 3.2. *An extendible array of varying dimensions is realizable, using the $\alpha\beta\gamma$-IndexArray scheme, by $\Re_\alpha = (\pi_\alpha, \Phi^*_\alpha, \Psi_\alpha)$, where π_α is the mapping function for the elements of the principal array, Φ^*_α is the mapping function for the elements in the $\alpha\beta\gamma$ index array and Ψ_α is an inverse subscript function for the array D_α.*

Theorem 3.3. *The complexity of computing the address of an arbitrary element in an extendible array of varying dimensions having n elements in d dimensions is $\tau(d, n; \Re^*_\alpha) = \Theta(d^2)$.*

Proof.
The computation of Φ^*_α is $\Theta(d)$, since every statement involves a constant number of elementary operations except for the evaluation of Ψ_α which takes at most $d-1$ comparisons. In computing π_α, the funtion Φ^*_α is computed d times to determine the block header address and a further d-1 times in the computation of sum of product terms. The total time to evaluate $\Re^*_\alpha$ then is a constant sum of $d * \Theta(d)$ terms which is $\Theta(d^2)$. ∎

Theorem 3.4. *The $\alpha\beta\gamma$-Index-Array scheme realizes a d-dimensional extendible array of variable dimensions having n elements using $\sigma(d, n; \Re^*_\alpha) = \Theta(n)$ storage locations.*

Proof.
The n elements of the principal array A^*_α are allocated in exactly n storage locations. Whenever the number of dimensions increases, the subscript range of the admitted dimensions becomes 2. Hence the size of each dimension is at least 2 except for one which is at most $n/2^{d-1}$. The maximum storage required for the $\alpha\beta\gamma$ index array is therefore $n * d^2/2^{d-1}$. The total storage requirement is at most $n + n * d^2/2^{d-1} + d$, which is $O(n)$.

The size of the $\alpha\beta\gamma$ index array is minimum if and only if the principal array is hypercubical. This implies that the size of each index range is $n^{1/d}$. Consequently the minimum storage requirement for the index array is $n^{1/d} * d^2$ and the minimum total storage required in the realization is $n + n^{1/d} * d^2 + d$, which is $O(n)$. The two results of the maximum and minimum storage requirements imply that $\sigma(d, n; \Re^*_\alpha) = \Theta(n)$. ∎

Corollary 3.1. *The $\alpha\beta\gamma$-Index-Array scheme, favours infinite family of shapes such that any array of n elements is spread over $\sigma(d, n; \Re^*_\alpha) \leq 3n$ locations.*

Proof.
The follows from the proof of Theorem 3.4. We have that the maximum storage required for any rectangular array of any shape is $\sigma(d, n; \Re^*_\alpha) = n + n * d^2/2^{d-1} + d$. This implies that $\sigma(d, n; \Re^*_\alpha) \leq 3n$. ∎

Note that the Index-Array scheme may be considered as a special case of the $\alpha\beta\gamma$-Index-Array scheme in which the dimensionality is bounded. Although this gives an improved worst case storage utilization, an element access cost is now $\Theta(d^2)$ instead of $\Theta(d)$. The $\alpha\beta\gamma$-Index-Array scheme is the only method known yet for managing storage for extendible arrays, such that the element access cost is independent of the number of elements in the array and further allows the number of dimensions to be varied.

4. Conclusion.

The results presented in the paper solves the open question of representing an extendible array of n elements in consecutive memory locations such that an array of any shape may be represented in Cn locations. We have not anly shown that $C \leq 3$, but have further given the solution for a more difficult problem where the number of dimensions is allowed to vary as well. The techniques presented establish a general framework for realizing extendible arrays.

Acknowledgement.

We appreciate the use of the computing facility of the Department of Systems and Computer Engineering, at Carleton University, for testing our algorithms. We also thank Nick Briggs for his assistance in the typing of this paper. This research is supported in part by the Natural Sciences and Engineering Research Council of Canada, grant No 102B-0000.

References.

[1] De Millo, R. A. Eisenstat, S. C. and Lipton, R. J. Preserving Average Proximity in Arrays. *CACM 21, 3 (March. 1978), 228–231.*

[2] Hansson, Ake, and Tärnlund, Sten-Ake. Ziz-zag Procedures for Memory Allocation and Retrieval of Dense Equilateral Arrays of Dynaically Varying Order. *Bit 16 (1976), 269–274.*

[3] Knott, G. D. Procedures for Managing Extendible Array Files. *Software-Practice and Experience 11 (1981), 63–84* .

[4] Knuth, D. E. The Art of Computer Programming. Vol. 1 : Fundamental Algorithms. *Addison-Wesley, Reading, Massachusetts, 1968.*

[5] Nievergelt, J., Hinterbeger, H. and Sevcik K. C. The Grid File: An Adaptatable Symmetric Multi-Key File Structure. *Eidgenössische Technishe Hochschule Zürich. Institut für Informatik (December 1981).*

[6] Otoo, Ekow J. Low Level Structures in the Implementation of the Relational Algebra. *Ph.D Dissertation (August 1983), McGill University, Montreal, Canada.*

[7] Otoo, Ekow J. A Digital Hashing Scheme for Dynamic Multiattribute Files. *School of Computer Science, Technical Report No SCS-TR-44, Carleton University, Ottawa.*

[8] Otoo, Ekow J. A Mapping Function for the Directory of a Multidimensional Extendible Hashing. *Proc. of the 10th International Conference on Very Large Data Bases (August 1984), Singapore.*

[9] Otoo, E. J. and Merrett, T. H. Dynamic Multipaging: A Multidimensional Structure for Fast Associative Searching. *School of Computer Science, Technical Report No SCS-TR-56, Carleton University, Ottawa, Canada. Submitted* .

[10] Otoo, E. J. and Merrett, T. H. A Storage Scheme for Extendible Arrays. *Computing 31 (1983), 1–9.*

[11] Ouksel, M. and Scheuermann, P. Storage Mapping for Multidimensional Linear Dynamic Hashing. *Proc. 2^{nd} Symposium on Principles of Database Systems, Atlanta, Georgia (1983), 90–105.*

[12] Rosenberg, A. L. Allocating Storage for Extendible Arrays. *J. ACM 21, 4 (October 1974), 652–670.*

[13] Rosenberg, A. L. Managing Storage for Extendible Arrays. *SIAM J. Comput. 4, 3 (September 1975), 287–306.*

[14] Rosenberg, A. L. and Stockmeyer, L. J. Hashing Schemes for Extendible Arrays. *J. ACM 24, 2 (April 1977), 199–221.*

[15] Standish, T. A. Data Structure Techniques. *Addison-Wesley, Reading, Massachusetts, 1980.*

[16] Stockmeyer, L. J. Extendible Array Realization with Additive Traversal. *Report RC-4578, IBM Thomas J. Watson Research Centre, Yorktown Heights, N. Y., 1973.*

DICTIONARIES ON BOUNDED DOMAINS

(Preliminary Version)

ROLF G. KARLSSON
Data Structuring Group
Department of Computer Science
University of Waterloo
Waterloo, Ont., Canada, N2L 3G1

ABSTRACT

This paper examines dictionary algorithms (search, insert, and delete) which use $O(n)$ space, when the keys are taken from the set $\{0,1,...,m-1\}$. We develop an $O(\sqrt{\log m})$ algorithm using $O(n)$ space, and prove that the result is optimal under a reasonable model.

INTRODUCTION

This paper deals with the problem of performing the operations search, insert, and delete on n keys from a bounded set $M=\{0,1,...,m-1\}$ when n is $\Omega(2^{\sqrt{\log m}})$ but $o(m)$, which includes an interesting and realistic range of values for m and n. The main result is a $\sqrt{\log m}$ worst case time lower bound when using $O(n)$ space under a reasonable model. In particular, the model encompasses the techniques used in two $O(\sqrt{\log m})$ algorithms, the first of which was given by Willard [5]. The second one is presented in this paper. The $O(\sqrt{\log m})$ time, $O(n)$ space algorithm introduced here, however, has a significant advantage compared with Willard's in that it is adapted to the distribution of the key values. Hence its average case is better. Our point of view is that a bounded domain is interesting and potentially practical. For instance, the common assumption that key values are arbitrary reals contrasts with the fact that computers have a finite precision; and in many applications this generality is unrealistic and perhaps even undesirable.

Denote $M^{(n)}$ as the subsets of M of cardinality n, and $S \in M^{(n)}$ as the current set of keys. A *dictionary* is a data structure which supports the following three set operations:

MEMBER(q,S): determine whether $q \in S$; if so, retrieve q.

INSERT(q,S): replace S by $S \cup \{q\}$, where $q \in M$.

DELETE(q,S): replace S by $S-\{q\}$, where $q \in M$.

The *space* complexity is the amount of storage, in $\log m$ bit words, required by the data structure. The *time* complexity is the worst case amount of time necessary to process any of the operations. The cost to create the data structure is measured by the *preprocessing* time.

All logarithms are given to base 2 unless otherwise specified.

RELATED WORK

For the MEMBER operation using minimal space, Yao [6] proved a general lower bound on the number of probes required in the worst case by fully adaptive search strategies, where each probing location can depend on the search key q, as well as all previous probing results. He showed that in the random access machine (RAM) model, $\lceil \log(n+1) \rceil$ probes are necessary if the size of the key domain, m, is greater than a Ramsey number $R(2n-1,n,n!)$ (which is at least exponential in n). On the other hand, if m is at most $2n-2$ then one probe suffices. Yao's lower bound depends on the fact that keys outside S may not be present in the table. He demonstrates that by going beyond the addressing power of a RAM and allowing arbitrary encodings, two probes suffice if $m \geq 2^{16n^2}$. For intermediate values he left the question open. Tarjan and Yao [4] presented a static $O(\log_n m)$ time algorithm using $O(n)$ space, after $\Theta(n^2)$ preprocessing. A subsequent paper by Fredman, Komlos, and Szemeredi [2] describes an $O(1)$ time, $O(n)$ space, static retrieval algorithm for all values of m. Their two-step hashing scheme allows arbitrary encoding of the set information in the tables. The preprocessing is $O(n^3 \log m)$. For the dynamic case, Willard [5] gave an $O(\sqrt{\log m})$ time algorithm using $O(n)$ storage. Another $O(\sqrt{\log m})$ algorithm, developed independently and concurrently, is presented in the next section.

A FAST SEARCH ALGORITHM

A partitioning $P^{(b)}$ of a domain M results in b (possibly empty) buckets with domains of equal size. We shall call $P^{(b)}$ a *b-way split*, or *split* for short. Consider a model in which we may perform

(i) a split of a bucket domain in b sub-buckets, where b is the number of keys in the bucket, and

(ii) a partition of an ordered set of keys in two ordered subsets.

The complexity measures are the number of such operations used and the number of words, each of $O(\log m)$ bits, required to represent the data structure.

The *distribution trie:* Given $S \in M^{(n)}$, we split M in n buckets, each of domain size m/n. This also partitions the key set by discriminating on the $\log(m/n)$ leading bits. We recursively continue the partitioning of the domain and its associated key set as long as a bucket contains more than some critical number of keys, cv, which will be determined below. Each trie leaf stores its cluster of at most cv keys in an AVL tree [1]. □

Theorem 1: A membership query in a distribution trie is answered in $O(\sqrt{\log m})$ time using $O(n)$ storage.

Proof: The algorithm to search for a key q in a distribution trie is:

SEARCH(M: domain, q: key)
if (b, the number of keys in bucket B of domain M) $\leq cv$ **then**
do BS(B, q) : search the AVL tree storing the set in B
else do begin M_k := the partition in $P^{(b)}$ including q/b;
SEARCH(M_k, q **mod** ($|M|/b$))
end

The worst case time using b-way split occurs when b is just above the critical value, since the bigger b is, the finer the split. Let b be approximately equal to cv and assume the b keys are clustering together through all the k trie levels. We get

$$b = m/b^k \tag{1}$$

as the resulting domain at the start of a binary search. To minimize the total time, we balance the algorithm so that

time on b-way split [SEARCH] = time on binary search [BS]

$$k = \log b \tag{2}$$

Substitute (2) into (1)

$$2^k = m/2^{k^2}$$

and get

$$k = O(\sqrt{\log m})$$

Note this leads to an $O(n\sqrt{\log m})$ storage scheme, which can be improved by doing a $c^i b$-way split, where c is a suitable constant less than 1, on the i^{th} trie level, instead of the straight b-way split. Then similar to above, we have

$$b = \frac{m}{(c^k b)^k}, \text{ and } \log b = k$$

which approximates to

$$2^{k^2} = m(1/c)^{k^2}$$

or

$$k^2 = k^2 \log(1/c) + \log m$$

giving a time estimate

$$k = (\log m/(1-\log(1/c)))^{1/2}$$

with space requirement

$$b \times (n/b) \times \sum_{i=1}^{k} c^i \leq \frac{c}{1-c} \times n$$

By choosing $c = 3/4$, we get $O(\sqrt{\log m})$ time, and $O(n)$ space.

□

We assume $n = \Omega(2^{\sqrt{\log m}})$ since otherwise a balanced binary tree, which operates in $O(\log n)$ time, would give a faster solution. In the restricted model that only allows the search operations b-way split and set partition, the result in theorem 1 is optimal. Later on in this paper we prove a $\Omega(\sqrt{\log m})$ bound in a somewhat stronger model. By contrast the faster algorithms by Tarjan and Yao [4] and Fredman et al.[2] operate in a more general model, the uniform RAM. Our scheme, however, has the advantage that it requires only $O(n\sqrt{\log m})$ preprocessing, and so can be extended to the dynamic environment.

An INSERT or a DELETE affects the cardinality in one bucket on $O(\sqrt{\log m})$ levels in the trie. We cannot afford to modify the split pattern on every update, instead we *amortize* the time needed to create new sub-structures over a sequence of update operations. We let the split change in powers of two. A node has an actual and an ideal number of keys stored, where the ideal number b gives the current split in the node's bucket. When the ratio between actual and ideal numbers are off by a factor of 2 (either way), we initiate the change. The details of the long amortizing argument can be found in [3]. We assert

Theorem 2: The distribution trie supports a dictionary on $M^{(n)}$ in $O(\sqrt{\log m})$ time using $O(n)$ space.

We may compare our routine favorably with Willard's q-fast trie [5]. Even though the local ratios between split and cardinality may increase beyond the interval [1/2,2] when using an $(3/4)^t b$-way split, the average run time for the distribution trie should nevertheless be much lower than for the q-fast trie. Willard's trie has a fixed branching of $2^{\sqrt{\log m}}$ at all internal nodes, while our branching is adapted to the key distribution. For a uniform distribution our asymptotic search time is $O(1)$ while Willard's algorithm requires $\Theta(\sqrt{\log m})$ time, due to a mandatory binary search of a $\Theta(2^{\sqrt{\log m}})$ set. The case is similar for updates, since insertion or deletion time is proportional to the trie height. For clustered distributions, our algorithm is asymptotically equivalent to Willard's trie.

A LOWER BOUND

Our objective is to prove an $\Omega(\sqrt{\log m})$ bound on dictionary algorithms for sets $S \in M^{(n)}$, stored in $O(n)$ words. We assume $\log n = \Omega(\sqrt{\log m})$ and $n = o(m)$, and consider a *segment graph model* given as follows:

i) A single-source directed graph G represents S, with the input node denoted the *root*.

ii) Each node represents a contiguous subset, a *segment*, of M, and the set of keys contained in the segment; in particular, the root represents M and S.

iii) A path between two nodes implies a reduction of the segment size represented at the two nodes.

iv) Each answer to a query is given by a path in G.

The graph nodes are *oblivious*, in the sense that each knows only the number of keys contained in their own segments, and that they are going to fan out. In this model, a search strategy may be composed of the following four operations:

1) A domain split, given as an edge in the graph.

2) A partition in two of an ordered key set at a node, performed irrespective of the set domain. For simplicity, we consider set bisection, which can only improve the worst case lower bound.

3) A bit encoding of the domain for a key set.

4) A domain split with a subset of keys represented by one key in the graph.

The important constraint to remember when employing these operations to compose a search strategy is that we may only use $O(n)$ storage space. The operations allow a finite number of strategies, defining data structures. We assert that the optimal data structure has a $\sqrt{\log m}$ worst case search time. Observe previous algorithms fit in this model.

Theorem 3: Under the oblivious node model, dictionary algorithms on sets $S \in M^{(n)}$ require $\Omega(\sqrt{\log m})$ time when using only $O(n)$ space.

Proof: The technique we use to establish a lower bound is to define an adversary by a state table. We consider a pair (s,k) where s denotes the segment size, and k the number of keys in the segment. The model operations define transitions from one state to another. Originally the algorithm is in state (m,n), and concludes in state (t_m,t_n), for $2^{\sqrt{\log m}} \leq t_m < m$, and $t_n \leq 2^{\sqrt{\log m}}$. Once in (t_m,t_n), a state $(t_{m'},1)$ can be reached in less than $\sqrt{\log m}$ set bisections.

We simplify the argument by considering a b-way split independent of the trie level i on which it occurs, instead of a $(3/4)^i b$-way split. In fact, this gives a stronger time lower bound.

The transitions are:

1) $(s,k) \Rightarrow (s/k,k)$; domain split

2) $(s,k) \Rightarrow (s,k/2)$; set bisection

3) $(s,k) \Rightarrow (1,1)$; bit map

4) $(s,k) => (s/(k \times c),k)$; domain split with reduction of sets of size c

This table is incomplete since we have to include the space complexity needed when choosing a transition. Alternative 3, superior in time complexity, requires $O(s)$ storage, which in significant cases is too much. Space dependency for the transitions are as follows:

1) Each segment requires $\Theta(k)$ space. We have maximized the domain reduction by splitting its size with the number of keys in the segment, where with oblivious nodes a uniform split is optimal. We cannot allow a higher split, since to store the splits will then require a total of $\omega(k \times (n/k)) = \omega(n)$ space per level.

2) Each set requires $\Theta(k)$ space independent of s.

3) Each map requires $\Theta(s/\log m)$ space independent of k.

4) Similar to case 1.

Observation 1: Transition 3 can only be used for encoding small domains, otherwise the space demand will be excessive. Even by straining the limits to allow an $n \log m$ domain split, squeezed into a bitmap of size $n \log m$, which is n words, we get no asymptotic effect on the optimal operation time.

Observation 2: Transition 4 is effectively equivalent to transition 1 in producing a fast search. The key cluster in 4 eventually has to be resolved by a binary search or a bitmap. Hence its size c should be at most $2^{\sqrt{\log m}}$, so as not to dominate the other cost factors in a search strategy. But this bound on c means that transition 4 is asymptotically equivalent to transition 1. We included the transition since it is used in Willard's q-fast trie.

The argument we present produces an $\Omega(\sqrt{\log m})$ bound on the membership query. In our model, this is equivalent to an $\Omega(\sqrt{\log m})$ bound on the query together with the updates. We make no a priori assumption on the key distribution. A balanced uniform domain split is therefore optimal, as model operation 1 by itself. We argue that the quickest way to get to a final state is by performing a sequence of 1-transitions, followed by a sequence of 2-transitions.

Choosing a 2-transition for a large set of $\omega(2^{\sqrt{\log m}})$ keys is wasteful since a domain spit there will bring us closer to a final state. We see this easiest if we repeatedly bisect such a large set $\sqrt{\log m}$ times. The resulting set is of $\omega(1)$ size. A repeated domain split on the other hand will reach a set of size $O(1)$ after $o(\sqrt{\log m})$ transitions, if we assume the key set stays large throughout the splits.

Choosing a 1-transition for a small set of $o(2^{\sqrt{\log m}})$ keys is also wasteful in the worst case. An adversary would cluster the keys so that a domain split does not produce a reduction of the key set. A repeated choice of transition 1 with a

clustered small set requires $\omega(\sqrt{\log m})$ iterations to reach a final state, since k is $\omega(\log b)$ if $b = o(2^{\sqrt{\log m}})$ and $m/b^k = O(b)$.

Hence we choose a sequence of domain splits until we reach a critical domain size. From there on we do a binary search. The two phases are balanced with a critical domain size $2^{\sqrt{\log m}}$, which produces a least total time of $O(\sqrt{\log m})$ in the worst case.

□

Our model includes the algorithmic techniques of both Willard's fast trie, and the distribution trie.

Corollary 4: The q-fast trie and the distribution trie are both optimal in a segment graph model with oblivious nodes.

Acknowledgement: I thank Ian Munro for his kind assistance, the distribution trie is a joint work with him.

REFERENCES

[1] Adel'son-Vel'skii, G.M. and Landis, E.M., "An Algorithm for the Organization of Information," *Doklady Akademia Nauk* USSR 146, 2 (1962), 263-266.

[2] Fredman, M.L., Komlos, J. and Szemeredi, E., "Storing a Sparse Table with O(1) Worst Case Access Time," *J. ACM 31,* 3 (July 1984), 538-544.

[3] Karlsson, R.G., "Algorithms in a Restricted Universe," Ph.D. dissertation in preparation (1984).

[4] Tarjan, R.E. and Yao, A.C.-C., "Storing a Sparse Table," *Comm. ACM 22,* 11 (Nov. 1979), 606-611.

[5] Willard, D.E., "Two Very Fast Trie Data Structures," *19th Allerton Conference* (1981), 355-363.

[6] Yao, A.C.-C., "Should Tables Be Sorted ?," *J. ACM 28,* 3 (July 1981), 615-628.

ON-LINE UPDATING OF DEGREE-CONSTRAINED MINIMUM SPANNING TREES

(preliminary version)

GREG N. FREDERICKSON
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

MANDAYAM A. SRINIVAS
Department of Computer Science
University of Rochester
Rochester, NY 14627

A degree-constrained minimum spanning tree for a graph is a spanning tree in which exactly s edges are incident on a specified vertex, for some s, and which is of minimum cost among all such trees. A data structure is presented for the problem of updating such a degree-constrained minimum spanning tree on-line when the cost of a single edge in the graph is modified. For a graph with m edges and n vertices, the algorithm is shown to require $O(U(m,n) + (\log m)^2)$ time and $O(m)$ space, where $U(m,n)$ is the time to update an unconstrained minimum spanning tree.

INTRODUCTION

Consider the problem of updating a minimum spanning tree of an underlying graph, which is modified repeatedly by having the cost of an edge changed. This problem has a very efficient solution, presented recently in [F]. Suppose however that a spanning tree is desired that has exactly s edges incident on one specified vertex, and is of minimum cost among all such trees. This problem models, for instance, the problem of constructing a tree-shaped computer network, in which a given computer has a constraint that exactly s links are to be attached to it. In this paper we present data structures for efficient on-line updating of such a degree-constrained minimum spanning tree. Our updating methods are considerably faster than those already known in the case when the communication graph is planar.

For an n-node planar graph, our update times are $O((\log n)^2)$. Thus a degree-constrained spanning tree can be updated within the same time bound as the best algorithm known for updating an unconstrained minimum spanning tree of a

planar graph [F]. The degree-constrained problem is a special case of a problem in which each edge is either red or green, and the desired spanning tree is of minimum cost among those with q red edges. The best algorithm known for updating such a red-green tree in a planar graph uses $O(2^{2\sqrt{\log n}}(\log n)^{3/2})$ time [FS], so that our algorithm is a considerable improvement in this case. Our algorithm can also be used for updating a degree-constrained tree in a general graph, but the resulting update time of $O(\sqrt{m})$ is no better than that for updating a red-green tree in a general graph [FS].

We briefly mention connections with earlier work. A number of papers have investigated update versions of geometric problems [WL, BS, OV]. Dynamic graph problems have also received recent attention [ST, F, H1]. Efficient algorithms for finding minimum spanning trees are presented in [CT,Y] and most recently in [FT]. The problem of finding a constrained spanning tree is a special case of a matroid intersection problem [L, W], for which the basic solution strategy is outlined in [G] and efficient algorithms for several specific matroids are presented in [GT].

PRELIMINARIES

Let G be a weighted undirected graph and v be the degree-constrained vertex. Let each edge incident on v be *green*, and all other edges be *red*. Thus a degree-constraint of s on vertex v implies that there will be s green and $q = n-s-1$ red edges in the desired tree. Let T_i be a minimum spanning tree for G with exactly i red edges, which is of minimum cost among all such trees. We assume that G has been augmented with edges as necessary so that a *green minimum spanning tree* T_0 and a *red minimum spanning tree* T_{n-1} exist.

Given a spanning tree T with at least one green edge, a *swap* (e, f) for T is an ordered pair of edges, where $e \in T$ is green, $f \notin T$ is red, and $T - \{e\} \cup \{f\}$ is a spanning tree. In this case f can be swapped in place of e, resulting in a spanning tree with one more red element. The *cost* $c(e, f)$ of a swap (e, f) is $c(f) - c(e)$. A swap (e, f) is a *best swap* for T if its cost is the minimum among all swaps for T. Ties should be broken lexicographically, ensuring that no two edges, swaps, or spanning trees will have the same cost.

A *swap sequence* $S_1, \ldots, S_{n-1}$ is a sequence of swaps such that $S_i = (e_i, f_i)$

is a best swap for T_{i-1}. As discussed in [G, GT], the desired tree T_q can be obtained by computing the swap sequence and then starting with T_0 and performing the first q swaps from the swap sequence. The choice of red edges can be restricted to T_{n-1}. In [FS], we established the following characterizations of the worst-case impact on T_q of a single edge cost change.

Theorem 1. [FS] Suppose a red edge r_t in T_q increases in cost. Let T_q' be the new degree-constrained minimum spanning tree. Then one of the following cases holds: (a) $T_q' = T_q$, (b) $T_q' = T_q - \{r_t\} \bigcup \{r_1\}$, for some particular red edge r_1, (c) $T_q' = T_{q+1} - \{r_t\} \bigcup \{g_1\}$, for some particular green edge g_1. Similar cases hold for other possible edge cost changes. □

Theorem 2. [FS] Let G be a weighted, undirected graph. If one edge in G changes in cost, then a degree-constrained spanning tree of minimum cost will have at most two edges replaced, one incident on the degree-constrained vertex, and the other not. □

THE DATA STRUCTURES

In this section we describe the data structures employed, and how they are constructed. Our basic strategy is to generate the data structures based on the recursive structure of Gabow and Tarjan's algorithm for finding a degree-constrained minimum spanning tree [GT]. To facilitate the updating, it is necessary to maintain a description of the local neighborhood around each vertex of the minimum spanning tree. We thus augment the structures derived from the Gabow and Tarjan algorithm with *incidence structures*, which are generalized heap structures formed using the topology tree structure from [F]. These data structures are constructed during a preprocessing stage when the original solution is computed.

The Gabow and Tarjan algorithm for the degree-constrained problem starts by computing the green minimum spanning tree T_0 and the red minimum spanning tree T_{n-1}. It then computes a set of swaps $M = \{(e,\mu(e)) \mid e$ is green and $(\mu(e),e)$ is the best swap for e in $T_0\}$. Given this *greedy swap set* M, it is easy to deduce some of the red edges in the tree T_q, by choosing those red edges in the q

cheapest swaps. The red edges that are found are then contracted, allowing the detection of the set of non-tree green edges which pair up with these contracted red edges in the swap sequence. The contractions cause certain of the green edges to become parallel in the graph, and for each set of parallel green edges, all but the edge of smallest cost are non-tree edges.

This process of generating M and identifying tree and non-tree edges is performed recursively on the contracted graph. Let d be the number of levels of recursion. It can be shown that $d = O(\log n)$. Since all the green edges in the graph are incident on a single vertex, a red edge can be adjacent to at most 2 green edges. This implies that a red edge can be the best swap for at most 2 green edges, and so the q least cost elements of M will contain at least $\lceil q/2 \rceil$ swaps of the swap sequence. Thus on each level the algorithm determines at least one half of the red tree edges remaining to be identified.

The preprocessing phase of our algorithm essentially executes the Gabow and Tarjan algorithm, and sets up our data structures for each level l of recursion, $l = 0, \ldots, d$. For each l, we organize a greedy swap set M_l. Let q_l be the number of red tree edges not yet identified just before level l. We partition the set M_l into two subsets: M_{l1}, containing the q_l smallest elements of M_l, and M_{l2}, containing rest of M_l. At each level, we use a balanced tree structure to implement *min*, *max*, *member*, *insert* and *delete* instructions on the sets M_{l1} and M_{l2}. In addition, we maintain a dictionary D_l^r of red edges included in (and a dictionary D_l^g of green edges excluded from) the solution as a result of swaps in M_{l1}.

For each vertex v_{lj} in the graph at level l, we will need to compute changes to the sets M_{l1} and M_{l2} due to edges being inserted or deleted. For this, we must to determine the least cost edge incident on each vertex v_{lj}, which represents a set V_{lj} of vertices in the original graph, and is associated with a tree-structured connected component of selected red edges from the original graph. We designate the structure used to extract the least cost edge incident on a vertex set V_{lj} at level l as the *incidence structure* of vertex set V_{lj}.

We use *topology trees* [F] to organize into a heap the set of red edges with exactly one endpoint in vertex set V_{lj}. In a topology tree, the vertices in V_{lj} are first partitioned into clusters. These basic clusters are themselves aggregated into larger clusters to form a multi-level partition of the vertex set. The topology tree is a

balanced tree whose structure is based on the topology of the clusters. Each internal node has at most four children, and all leaves are at the same depth. A leaf represents a basic vertex cluster, and an internal node represents the aggregate cluster obtained by the union of the clusters represented by its children. A topology tree on n vertices supports *split* and *join* operations in $O(\log n)$ time, and a *min* operation in constant time.

The construction procedure for topology trees requires that each vertex in the underlying tree-structured connected component should have degree no greater than 3. A modified version of a well-known transformation in graph theory [H, p. 132] may be used to generate such a tree from the given tree at each level l. For each vertex u of degree $d > 3$, where $w_1, \ldots, w_d$ are the vertices adjacent to u, replace u with new vertices $u_1, \ldots, u_d$. Add red edges $\{(u_i,u_{i+1}) \mid i=1, \ldots, d-1\}$ each of cost zero, and replace the edges $\{(w_i,u) \mid i=1, \ldots, d\}$ with $\{(w_i,u_i) \mid i=1, \ldots, d\}$, of corresponding costs. Observe that the degree-constrained vertex v does not need to be transformed since it does not need an incidence structure, and that the transformation introduces $O(m)$ new vertices per level, but leaves the greedy swap sets unaffected.

The incidence structures at any one level l will have a total of $O(m)$ elements, and so $O(m \log m)$ space may be required in the obvious implementation when all levels are considered. We show how to reduce this to $O(m)$ by using the incidence structures at level l to generate those at level $l+1$. Consider a vertex $v_{l+1,j}$. Its associated connected component of selected red edges is the union of component edges selected at level l and edges selected at lower levels. We can construct the incidence structure for $v_{l+1,j}$ by a sequence of *join* operations involving the edges selected at level l and the level l incidence structures corresponding to each vertex v_{lk} on which these selected edges are incident. Performing a *join* on an incidence structure involves visiting and modifying a certain number of nodes in the topology tree. We will create a new level $l+1$ node for each level l node visited or modified, thus effectively reusing unaltered subtrees of the level l incidence structure. We show that only $O(m)$ new nodes are created at all levels.

Consider the forest of topology trees resulting at level $l+1$. If b red edges have been added at this level, then the forest will have new nodes along essentially b paths from the leaves. Thus there will be $O(b)$ new nodes in each of the lowest $\lceil \log(mb) \rceil$ levels of the forest. In addition, every node at higher forest levels could

be new. However, in any forest of topology trees consisting of m nodes in all, the number of nodes at heights greater than $\lceil \log a \rceil$ is $O(ma)$. Thus there can be at most $O(b)$ new nodes at levels other than the lowest $\lceil \log(mb) \rceil$ in the forest. This gives $O(b+b\log(mb))$ new nodes altogether. When we consider all levels of the data structure, the space is therefore bounded by $\sum_{i=1}^{\lceil \log m \rceil} c(b_i + b_i \lceil \log(mb_i) \rceil)$. Since at least half the remaining red edges are determined at each level, $b_i \leq m2^{i+1}$. The space requirement is therefore at most $c \sum_{i=1}^{\lceil \log m \rceil} (m2^{i-1}(1+\log 2^{i-1}))$ $= c \sum_{i=1}^{\lceil \log m \rceil} (im2^{i-1}) = O(m)$. A reference count scheme can be used to return nodes deleted during the update process to the storage pool, so that the number of nodes will remain $O(m)$ after each update.

Theorem 3. The data structure for on-line updating of a degree-constrained spanning tree can be set up using $O(T(m,n) + n \log n)$ time and $O(m)$ space, where $T(m,n)$ is the time to construct a minimum spanning tree.

Proof. The time to execute Gabow and Tarjan's algorithm is $O(T(m,n) + n \log n)$. Since the additional time to set up the incidence structures is $O(m)$, the time bound follows. The space bound is discussed above. □

Currently the best bound for $T(m,n)$ is $O(m\beta(m,n))$ [FT], where $\beta(m,n)$ is a certain very slowly growing function.

THE UPDATE ALGORITHM

We next describe the update process. Suppose a red edge $r_t = (w,x)$ changes in cost. (Handling a cost change for a green edge is similar). The red minimum spanning tree is first recomputed. As a worst case, suppose the edge (w,x) is replaced by a new edge (y,z) in this tree. The entries of the incidence structures for the vertices w,x,y and z must be adjusted. Let A_l be the set of vertices $\{x \mid x$ is at level l and the minimum value of the incidence structure of x has been modified$\}$. Thus $A_0 \subseteq \{w,x,y,z\}$ initially. The incidence structures of x are updated for all

$x \in A_0$. Then the data structures representing the greedy swap sets M_{01} and M_{02} must be updated to reflect the movement of a swap from M_{01} to M_{02}, and vice versa. We update the set of solution edges determined at level 0 in our dictionary D_0^S. We likewise update D_0^g.

We next describe how the changes affect the incidence structures. A red edge entering the solution bridges two connected components, and a departing red edge splits a component into two. Let (y,z) be an entering edge. We need to find the components containing y and z. Let the level 1 vertices corresponding to these components be w and x. We then need to *join* the incidence structures of w and x. As a second case, let (w,x) be an edge leaving D_0. Let y be the level 1 vertex corresponding to the component containing both w and x at level 0. We need to *split* y's incidence structure at the edge (w,x). The set A_1 will contain those vertices whose incidence structures have had their entries modified due to *split*'s and *join*'s at level 0.

At deeper levels, the update algorithm repeats the above process for each element in the set A_l. After the data structure has been updated, we need to report the net change in the solution T_q. Therefore during the update process we generate, at each level l, sets representing the difference between the old and the new partial solutions. We compute the new red tree edges identified by the end of level l, which had not been identified in the first l levels before the update, and the new red non-tree edges which had not been identified earlier. New green tree and non-tree edges are similarly processed. The net change in the solution can then be deduced from these sets at level d.

We now analyze the efficiency of the update algorithm. We first prove a lemma which shows that at each level l, the portion of the swap sequence being considered does not change very much. Recall that q_l is the number of red edges needed to be swapped in before level l is handled. Let q_l' be similarly defined after a red edge has increased in cost. The lemma may be shown to also hold for the the other cases of possible edge cost changes.

Lemma 1. For each l, one of the following cases holds:

0. $q_l = q_l'$ 1. $q_l - q_l' = 1$ 2. $q_l' - q_l = 1$

Proof. We note that Gabow and Tarjan's algorithm discovers red edges in the swap sequence strictly from left to right. The proof is by induction on l. Case 0 holds initially, when $l = 0$. For $l > 0$, we consider the following cases.

Case 0 after level $l-1$: Let e_l and e_l' be the number of red edges added to the spanning tree at level l. If $e_l'-e_l = 0$, then case 0 will hold after level l, so let us suppose that $e_l' > e_l$; the argument is similar when $e_l' < e_l$. Since more edges are determined from M_l' than from M_l, there must be some red edge r_x which is shared by two green edges g_a and g_b in M_l but not in M_l', *i.e.*, if swaps (g_a,r_x) and (g_b,r_x) appear in M_l, then M_l' has swaps (g_a,r_a) and (g_b,r_b), with $r_a \neq r_b$. We will refer to an edge such as r_x as a *split* edge. Let $z = q - q_l$ represent the number of red edges discovered by the end of level $l-1$. From Theorem 1, we know that there is at most one red edge in both $T_z' - T_z$ and $T_z - T_z'$. Thus there can be at most one split red edge in M_l, and case 1 will hold after level l.

Case 1 after level $l-1$: Evidently, there is a split red edge r_x in some M_j, $j < l$, with swaps (g_a,r_a) and (g_b,r_b) appearing in M_j'. Without loss of generality, let $c(g_a) < c(g_b)$. If M_l has a split red edge r_y, we can show that r_y must be either r_a or r_b. But then g_a would have to be one of the edges sharing r_y in M_l, and g_a cannot appear in M_l' since it has already appeared in M_j'. Thus M_l has no split red edge and M_l' has either 0 or 1 split red edge. Let s_l be the number of shared red edges in the first q_l swaps of M_l. Then $e_l = q_l - s_l$, and from the previous discussion, $s_l' - s_l = 0$ or 1. Since $q_{l+1} = q_l - e_l$, $q_{l+1}' - q_{l+1} = (q_l' - e_l') - (q_l - e_l)$ $= s_l' - s_l = 0$ or 1, resulting in case 0 or case 2, respectively, after level l.

Case 2 after level $l-1$: Similar to case 1. □

Theorem 4. The on-line update problem for degree-constrained minimum spanning trees can be solved in $O(U(m,n) + (\log m)^2)$ time, where $U(m,n)$ is the time to update a minimum spanning tree.

Proof. From Lemma 1 we conclude that at some particular level, the selected red edges could come from the swaps $(S_i, \ldots, S_j)$ before the edge cost change, and from the swaps $(S_{i'}, \ldots, S_{j'})$ after the change, where $i' = i \pm 1$ and $j' = j \pm 1$. Theorem 1 can be applied to show that the symmetric difference of these two sets of swaps can contain no more than 4 red edges. Thus the number of incidence structures affected at any level is constant, and the number of *splits* and *joins* per level is also a constant. Since each *split* or *join* takes $O(\log m)$ time and there are $O(\log n)$ levels, the procedure runs in time $O((\log m)^2)$ in addition to the time required to update a minimum spanning tree. □

The best bounds for $U(m,n)$ are currently $O(\sqrt{m})$ for general graphs and $O((\log m)^2)$ for planar graphs [F]. Thus we are able to update a degree-constrained minimum spanning tree in time equal to that for an unconstrained minimum spanning tree, and the update is especially efficient for planar graphs.

REFERENCES

[BS] J. L. Bentley and J. B. Saxe, *Decomposable searching problems I. Static-to-dynamic transformation* J. Algorithms 1 (1980) 301-358.

[CT] D. Cheriton and R. E. Tarjan, *Finding minimum spanning trees*, SIAM J. Comput. 5 (1976) 724-742.

[F] G. N. Frederickson, *Data structures for on-line updating of minimum spanning trees, with applications*, to appear in SIAM J. Comput.

[FS] G. N. Frederickson and M. A. Srinivas, *Data structures for on-line updating of matroid intersection solutions*, Proc. 16th ACM Symp. on Theory of Computing, Washington, D.C. (1984) 383-390.

[FT] M. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, to appear in Proc. 25th Symp. on Foundations of Computer Science, Singer Island, Florida (1984).

[G] D. Gusfield, *Matroid optimization with the interleaving of two ordered sets*,

Disc. Appl. Math. 8 (1984) 41-50.

[GT] H. N. Gabow and R. E. Tarjan, *Efficient algorithms for a family of matroid intersection problems*, J. Algorithms 5 (1984) 80-131.

[H] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

[H1] D. Harel, *On line maintenance of the connected components of dynamic graphs*, manuscript (1982).

[L] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[OV] M. H. Overmars and J. van Leeuwen, *Maintenance of configurations in the plane*, J. Comp. Sys. Sci. 23 2 (1981) 166-204.

[ST] D. D. Sleator and R. E. Tarjan, *A data structure for dynamic trees*, J. Comp. Sys. Sci., 26 (1983), 362-391.

[W] D. J. A. Welsh, *Matroid Theory*, Academic Press, New York, 1976.

[WL] D. E. Willard and G. Lueker, *A transformation for adding range restriction capability to data structures*, to appear in J. ACM.

[Y] A. C. Yao, *An $O(|E|\log\log|V|)$ algorithm for finding minimum spanning trees*, Inf. Proc. Lett. 4 (1975) 21-23.

On The Probable Performance of Graph Coloring Algorithms

Jonathan S. Turner
Computer Science Department, Campus Box 1045
Washington University, St. Louis, MO 63130
jst@wucs.UUCP

ABSTRACT

We define a natural probability distribution over the set of k-colorable graphs on n vertices and study the probable performance of several algorithms on graphs selected from this distribution. The main results are listed below.

1. We describe an algorithm to determine if a given n vertex graph is k-colorable, which runs in time $O(n + m \log k)$, where m is the number of edges. We show that this algorithm can successfully identify almost all random k-colorable graphs for constant or slowly growing values of k.
2. We show that an algorithm proposed by Brélaz, and justified on experimental grounds can successfully k-color almost all random k-colorable graphs for constant or slowly growing values of k. We also describe an efficient implementation, which runs in time $O(m \log n)$.
3. We show that the performance of the well-known greedy algorithm for graph coloring is relatively poor for random k-colorable graphs.

In addition to the analytical results, we present experimental data which provide more detailed information on the performance of these algorithms.

1. Introduction

Let $G = (V, E)$ be a simple undirected graph. A k-coloring of G is a mapping $c : V \rightarrow \{1, 2, \ldots, k\}$; c is a *proper coloring* if $c(u) \neq c(v)$ for all $\{u, v\} \in E$. The *chromatic number* of G, denoted $\chi(G)$, is defined as the smallest positive integer k for which a proper k-coloring exists. The *graph coloring problem* is to determine for a given graph G and an integer k, if $\chi(G) \leq k$.

The graph coloring problem has a long, interesting history and arises in a variety of applications. Karp [7] showed that the problem is NP-complete. Stockmeyer [9,3] strengthened this by showing that it remains NP-complete for any fixed $k \geq 3$. This has led many researchers to seek approximation algorithms capable of producing colorings that don't use too many extra colors. Garey and Johnson [4] proved that unless P = NP, no polynomial time approximation algorithm can guarantee the use of less than $2\chi(G)$ colors. Furthermore, Johnson [6] showed that for many popular heuristics, there are 3-colorable graphs on n vertices for which the heuristics require $\Theta(n)$ colors. Johnson also described a new algorithm using at most $O(n/\log n)$ colors on any 3-colorable graph. This stood as the best worst-case result for graph coloring until recently when Wigderson [11] discovered an algorithm that colors any 3-colorable graph using at most $3\lceil\sqrt{n}\rceil$ colors and any k-colorable graph using at most $2k\lceil n^{1-1/(1-k)}\rceil$ colors.

The disappointing nature of the worst-case results for graph coloring suggests that probabilistic analysis may provide a more effective way of evaluating candidate algorithms. Grimmet and McDiarmid [5] took the first step in this direction by showing that for almost all graphs on n vertices, $\chi(G) \geq (1-\epsilon)n/(2\log_{1/(1-p)} n)$, where p is a fixed edge probability in the usual random graph model, [1] and ϵ is any positive constant. The also showed that a well-known greedy heuristic uses $\leq (1+\epsilon)n/\log_{1/(1-p)} n$ colors.

[1] In the usual random graph model, edges are generated independently with probability p between each pair of vertices. We say that a property holds for *almost all* random graphs if the probability of the property holding approaches one as $n \rightarrow \infty$.

Grimmet and McDiarmid's results are interesting for what they tell us about random graphs; it's less clear what they tell us about the merits of the greedy heuristic. The naive conclusion one can draw is that the greedy algorithm is a good one for graph coloring. A less obvious, but perhaps more accurate interpretation is that these results cast doubt on the usefulness of a probabilistic analysis based on the usual random graph model for comparing graph coloring algorithms. They suggest that the usual model is too 'easy' a distribution, since it makes even the most simple-minded algorithm look good. In order to obtain meaningful comparative information, we should try to select a more difficult probability distribution, one that poses some challenges for candidate algorithms to overcome. The probability distribution described in the next paragraph and used throughout this paper was motivated in part by these considerations. Another factor motivating this choice is touched on briefly at the end of section 2.

Let n, k be positive integers, $0 < p < 1$ and let $G = (V, E)$ be the graph defined by the following experiment.

- Let $V = \{1, \ldots, n\}$.
- For each $u \in V$ let $c(u)$ be a random integer in $[1, k]$.
- For each pair $u, v \in V$ such that $c(u) \neq c(v)$, include the edge $\{u, v\}$ in E with probability p.

The probability distribution defined by this experiment is denoted $X_n(k, p)$ and the notation $G \in X_n(k, p)$ means that G is a random graph generated in this way.

In section 2, we present a coloring algorithm, which for constant p and k growing slowly with n, finds a k-coloring for almost all graphs $G \in X_n(k, p)$, and we describe an efficient implementation using a novel data structure. In section 3, we discuss an algorithm due to Brélaz [2] and show that under the same conditions it has similar performance. We also describe an implementation of Brélaz's algorithm with running time $O(m \log n)$ for graphs on n vertices and m edges. (Brélaz claimed only an $O(n^2)$ running time.) In section 4, we study the performance of the greedy algorithm for graphs in $X_n(k, p)$; this study reveals performance that is surprisingly poor in light of [8].

2. Recognizing k-Colorable Graphs

Define a *partial coloring* of a graph $G = (V, E)$ to be a mapping $c : V \to [0, n]$. The algorithms we will study start by constructing the partial coloring defined by $c(x) = 0$ for all $x \in V$ and then attempt to convert this to a complete proper coloring. Given a partial coloring c, we can define for each vertex x, a set $avail_c(x) = \{i \mid 1 \leq i \leq n \wedge (\{x, y\} \in E \Rightarrow c(y) \neq i)\}$. If x is currently uncolored ($c(x) = 0$), $avail_c(x)$ is the set of colors that are available for coloring x. We will write $avail(x)$ without the subscript whenever the coloring function is clear from the context.

Our first algorithm attempts to find a k-coloring of a graph $G = (V, E)$, where k is assumed to be an input parameter. We will show that it succeeds with high probability for random k-colorable graphs. The algorithm starts by finding a k-clique, coloring each of its vertices with a distinct color in $[1, k]$, and then repeatedly applies the following rule.

> *Coloring Rule 1.* Select an uncolored vertex x for which $|avail(x) \cap [1, k]| = 1$ and let $c(x) = \min avail(x)$.

We refer to this as the *no choice algorithm* since it succeeds only if it can color all the vertices without making any arbitrary choices. The algorithm can fail to produce a k-coloring if it is unable to find a k-clique or if at some point $|avail(x) \cap [1, k]| \neq 1$ for all uncolored vertices x. We will show that when k is not too large, the no choice algorithm succeeds with high probability. First however, we give a more detailed description of the algorithm.

A program implementing the no choice algorithm is shown in Figure 1. (The algorithmic notation is adapted from Tarjan [10].) Vertices are represented by integers in $[1, n]$ and the graph is represented by an array of vertex sets called *neighbors*. For each vertex x, *neighbors*(x) is a list containing all vertices adjacent to x in increasing order. Vertices that are ready to be colored are placed in a queue. Each iteration of the algorithm's main loop removes a vertex from the queue, colors it, then examines its neighbors, adding them to the queue if possible. Initially each vertex is assigned a color of -1. When a vertex is added to the queue, its color is changed to 0. The subroutine shown in Figure 2 is used to find

```
function bit nochoice(integer k,n, graph neighbors, modifies array c);
        integer i,nc; vertex x,y; list Q; set X;
        array[1..n] of set avail;
        for x ∈ [1..n] → c(x) ← −1; avail(x) ← {1,...,k}; rof;
        X ← clique(k,n,neighbors);
        if X = → return false fi;
        i ← 1;
        for x ∈ X → c(x) ← i; i ← i + 1; rof;
        Q ← [ ];
        for x ∈ X →
                for y ∈ neighbors(x) →
                        avail(y) ← avail(y) − c(x);
                        if c(y) = −1 and |avail(y)| = 1 →
                                Q ← Q & [y];
                                c(y) ← 0;
                        fi;
                rof;
        rof;
        nc ← k;
        do Q ≠ [ ] →
                x ← Q(1); Q ← Q[2..];
                if |avail(x)| ≠ 1 → return false fi;
                let i ∈ avail(x); c(x) ← i; nc ← nc + 1;
                for y ∈ neighbors(x) →
                        avail(y) ← avail(y) − c(x);
                        if c(y) = −1 and |avail(y)| = 1 →
                                Q ← Q & [y];
                                c(y) ← 0;
                        fi;
                rof;
        od;
        return nc = n;
end;
```

Figure 1: Program Implementing the No Choice Algorithm

a clique. The *clique* program runs in linear time, if the set S is implemented as a sorted list of vertices. The key to efficient implementation of the main program is the data structure used to represent the sets $avail(x)$. The simplest approach is to use a bit vector for each set. This leads to an $O(kn+m)$ running time for a graph with n vertices and m edges. We can improve on this by using a special variety of binary search tree described below. (Note that a standard search tree won't help here since initializing n search trees to represent the set $\{1,\ldots,k\}$ takes $\Omega(kn \log k)$ time.)

We define a *shrinking set* to be an abstract data type representing a set of positive integers on which the following operations can be performed.

makeset(lo, hi) Return a new set consisting of the integers in the interval $[lo, hi]$.

select(s) Return an arbitrary element from s.

selectmin(s) Return the smallest element in s.

delete(x, s) Delete the integer x from s.

The operations on shrinking sets are defined in terms of another abstract data structure, which we call an *interval set*. An interval set represents a set of disjoint intervals on the positive integers on which the following operations are defined.

makeintervalset(i) Return a new set consisting of the interval i.

```
set function clique(integer k, n; graph neighbors);
        set S, K;
        S ← {1, ..., n};
        K ← ∅;
        do S ≠ ∅ →
                let x ∈ S;
                K ← K ∪ {x};
                S ← S ∩ neighbors(x);
                if |K| = k → return K fi
        od;
        return ;
end;
```

Figure 2: Subroutine for Finding a Clique

```
i ← member(x, s);
if i ≠ [] →
        deleteinterval(i, s);
        if i.lo < x → insertinterval([i.lo, x − 1], s) fi;
        if i.hi > x → insertinterval([x + 1, i.hi], s) fi;
fi;
```

Figure 3: Program Fragment Implementing the Delete Operation

member(x, s)	Return the interval in s that contains the integer x. If there is no such interval, return [].
select(s)	Return an arbitrary integer contained in some interval in s.
selectmin(s)	Return the smallest integer contained in some interval in s.
insertinterval(i, s)	Insert the interval i in s (i is assumed to be disjoint from intervals already in s).
deleteinterval(x, s)	Delete the interval i from s.

An interval set can be implemented efficiently using any standard balanced search tree structure. Each node of the search tree represents an interval. This yields an $O(\log n)$ running time per operation, where n is the number of intervals in the set. The operation *makeset*(lo, hi) on a shrinking set is implemented simply as *makeintervalset*$([lo, hi])$ on the underlying interval set. The *select* and *selectmin* operations on a shrinking set are implemented as the corresponding interval set operations. Finally, the operation *delete*(x, s) on a shrinking set is implemented by the program fragment in Figure 3. Thus, all the operations on a shrinking set can be implemented to run in $O(\log k)$ time, where k is the size of the set when it is initialized. These observations yield the following theorem.

Theorem 2.1: The no choice algorithm can test a graph for k-colorability in $O(n + m \log k)$ time, where n is the number of vertices and m the number of edges.

We now address the question of effectiveness for the no choice algorithm. First, some definitions.

Define $\lambda_n(c) = -\frac{\ln n}{\ln c}$. Note that $\lambda_n(c) > 0$ when $0 < c < 1$ and $n > 1$, $c^{\lambda_n(c)} = \frac{1}{n}$ and $\lim_{n\to\infty} \lambda_n(c) = \infty$ for fixed $c \in (0, 1)$. We will usually write $\lambda(c)$ instead of $\lambda_n(c)$.

In the remainder of this section $G = (V, E) \in X_n(k, p)$, where $0 < \epsilon < 1$, $0 < p < 1$ are fixed, $n > 0$, $0 < k \leq (1 - \epsilon)\lambda(p)$. We also let $V_1, \ldots, V_k$ be the partition of V used in the generation of G and $c(v) = i$ for each $v \in V_i$. Also, $n_i = |V_i|$ for $1 \leq i \leq k$ and $m = \min_{1 \leq i \leq k} n_i$.

We can now state the main theorem.

Theorem 2.2: For almost all G, the no choice algorithm produces a complete k-coloring.

Since the no choice algorithm makes no arbitrary decisions with the exception of coloring the initial clique, the coloring it produces is unique.

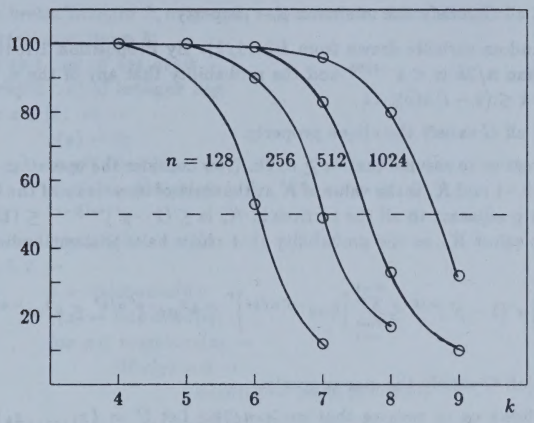


Figure 4: Success Rate of No Choice Algorithm for Graphs in $X_n(k, .5)$

Corollary 2.1: Almost all G are uniquely k-colorable.

Theorem 2.2 is proved in three steps. First, we define a class of graphs called *normal* graphs. Then, we show that the no choice algorithm succeeds for all normal graphs. Finally, we show that almost all G are normal.

We call G *normal* if it satisfies the following five properties and $n \geq \max\{k^2\lambda(1-p)/(1-\epsilon), (2k^2\lambda(1-p)/(1-\epsilon))^{1/\epsilon}\}$.

minimum size property — $m \geq n/2k$.

clique property — the clique finding algorithm of Figure 2 succeeds on G.

star property — for every set $\{x_1, \ldots, x_k\}$ with $x_j \in V_j$, each set V_i contains at least $(1-\epsilon)n^\epsilon/k$ vertices with $k-1$ neighbors in $\{x_1, \ldots, x_k\}$.

subset property — there is an edge joining every pair U_i, U_j $(i \neq j)$, where U_i is any subset of V_i with at least $k\lambda(1-p)$ vertices.

degree property — every vertex x has at least $(1-\epsilon)mp$ neighbors in each set V_i except $V_{c(x)}$.

It is convenient to think of the no choice algorithm as proceeding by a series of discrete steps. We now describe a version of it in which these steps are made explicit.

1. Find a k-clique $x_1, \ldots, x_k$ and let $c(x_i) = i$ for $1 \leq i \leq k$.
2. For $1 \leq i \leq k$, find a set U_i of at least $n^\epsilon/(1-\epsilon)k$ uncolored vertices adjacent to each vertex in $\{x_1, \ldots, x_k\} - \{x_i\}$. Let $c(y) = i$ for all $y \in U_i$.
3. For $1 \leq i \leq k$, find a set W_i of at least $n_i - k\lambda(1-p)$ vertices, such that every vertex $y \in W_i$ is adjacent to some vertex in each U_j for which $j \neq i$. Let $c(y) = i$ for all $y \in W_i$.
4. For each uncolored vertex y, such that $|avail(y) \cap [1,k]| = 1$ let $c(y) = \min avail(x)$.

It's easy to see that if the above algorithm succeeds in finding a k-coloring then the original no choice algorithm does also. While less obvious, it is straightforward to verify that if G is normal, then the algorithm succeeds. The lemmas that follow show that almost all G are normal, which in turn yields the theorem.

The following proposition (Angluin and Valiant [1]) is used in the proofs of several of the lemmas. Let $B(n,p)$ denote the binomial distribution. By definition, if $x \in B(n,p)$ then $P(x=k) = \binom{n}{k}p^k(1-p)^{n-k}$.

Proposition 2.1: If $x \in B(n,p)$ then for any α, $0 < \alpha < 1, P(x \leq (1-\alpha)np) < e^{-\alpha^2 np/2}$ and $P(x \geq (1+\alpha)np) < e^{-\alpha^2 np/3}$.

Lemma 2.1: Almost all G satisfy the minimum size property.

proof. Each n_i is a random variable drawn from $B(n, 1/k)$. By Proposition 2.1, the probabilty that a particular n_i is less than $n/2k$ is $< e^{-n/8k}$ and the probability that any of the n_i is less than $n/2k$ is $< ke^{-n/8k} \to 0$, since $k \leq (1-\epsilon)\lambda(p)$. □

Lemma 2.2: Almost all G satisfy the clique property.

proof. Lemma 2.1, allows us to assume that $m \geq n/2k$. Now consider the operation of the clique finding algorithm. If $0 \leq r \leq k-1$ and K_r is the value of K at the start of iteration r of the loop, the probability that there is no vertex y adjacent to all the vertices in K_r is $\leq (1-p^r)^{m(k-r)} \leq (1-p^r)^{n/k}$. There are at most $\binom{k}{r} n^r$ ways to select K_r, so the probability that *clique* halts without finding a k-clique is

$$\leq \sum_{r=0}^{k-1} \binom{k}{r} n^r (1-p^r)^{n/k} \leq \sum_{r=1}^{k-1} \left[kne^{-p^r n/kr} \right]^r \leq k^2 ne^{-p^k n/k^2} \leq k^2 ne^{-n^\epsilon/k^2} \to 0 \qquad \square$$

Lemma 2.3: Almost all G satisfy the star property.

proof. Lemma 2.1, allows us to assume that $m \geq n/2k$. Let $U = \{x_1, \ldots, x_k\}$ be any vertex set with $x_i \in V_j$ and let s_i be the number of vertices adjacent to every vertex in $U - \{x_i\}$. Clearly, $s_i \in B(n_i, p^{k-1})$. Since $n^\epsilon/k \leq n_i p^{k-1}$, $P(s_i \leq (1-\epsilon)n^\epsilon/k) \leq P(s_i \leq (1-\epsilon)n_i p^{k-1})$, which by Proposition 2.1 is,

$$< e^{-\epsilon^2 n_i p^{k-1}/2} \leq e^{-\epsilon^2 np^k/2k} \leq e^{-\epsilon^2 n^\epsilon/2k}$$

Since there are at most n^k different choices for U, the probability that G does not satisfy the star property is

$$< kn^k e^{-\epsilon^2 n^\epsilon/2k} = kn^{k-(\epsilon^2/2)(n^\epsilon/k \ln n)} \to 0 \qquad \square$$

Lemma 2.4: Almost all G satisfy the subset property.

proof. Lemma 2.1, allows us to assume that $m \geq n/2k$. For $1 \leq i \leq k$, let $U_i \subseteq V_i$ with $|U_i| = r \geq k\lambda(1-p)$. The probability that there is no edge joining a particular pair U_i, U_j is $(1-p)^{r^2}$. Since there are at most $\binom{n}{r}^k$ ways to choose the U_i, the probability that G does not satisfy the subset property is

$$\leq \binom{k}{2} \binom{n}{r}^k (1-p)^{r^2} \leq k^2 \left(\frac{en}{r}\right)^{kr} (1-p)^{r^2} \leq k^2 \left(\frac{en}{r}(1-p)^{r/k}\right)^{kr}$$

$$\leq k^2 \left(\frac{en}{k\lambda(1-p)}(1-p)^{\lambda(1-p)}\right)^{kr} \leq k^2 \left(\frac{e}{k\lambda(1-p)}\right)^{k^2\lambda(1-p)} \to 0 \qquad \square$$

Lemma 2.5: Almost all G satisfy the degree property.

proof. Lemma 2.1, allows us to assume that $m \geq n/2k$. Let $d_i(x)$ be the number of neighbors vertex x has in V_i. Clearly, $d_i(x) \in B(n_i, p)$ for $x \notin V_i$. By Proposition 2.1,

$$P(d_i(x) \leq (1-\epsilon)mp) \leq P(d_i(x) \leq (1-\epsilon)n_i p) < e^{-\epsilon^2 n_i p/2} \leq e^{-\epsilon^2 np/2k}$$

So the probability that G does not satisfy the degree property is $\leq kne^{-\epsilon^2 np/2k} \to 0$. □

This completes the proof of Theorem 2.2.

A series of experiments were run to provide more detailed information on the performance of the no choice algorithm. One hundred random graphs in $X_n(k, .5)$ were generated for each of several values of n and k. The no choice algorithm was then run on each graph. [2] The results are summarized in Figure 4. For each value of n and k the figure shows the number of graphs for which a k-coloring was constructed. The figure shows that the algorithm works well when k is small, but as k gets larger, its performance deteriorates abruptly. This is consistent with the analysis given above. As n increases, the breakdown point also increases. Let $\beta_n(p)$ be the smallest k for which the probability of success on graphs in $X_n(k,p)$ is less than 1/2. We can estimate $\beta_n(p)$ by observing where the curves in Figure 4

[2] The experiments actually involved a slightly different version of the no choice algorithm. This version uses a greedy heuristic to find the initial clique. When selecting the next vertex for inclusion in the clique, this heuristic always selects a vertex of maximum degree from the set of eligible vertices.

```
procedure brelaz(integer k, n, graph neighbors, modifies array c);
        vertex x, y; heap h;
        array[1..n] of set avail;
        array[1..n] of integer deg;
        for x ∈ [1..n] →
                c(x) ← 0;
                avail(x) ← {1, ..., n};
                deg(x) ← |neighbors(x)|;
        rof;
        h ← makeheap({1, ..., n});
        do h ≠ →
                x ← deletemin(h);
                c(x) ← min avail(i);
                for y ∈ neighbors(x) →
                        if c(y) = 0 →
                                avail(y) ← avail(y) − c(x);
                                deg(y) ← deg(y) − 1;
                                siftup(y, h);
                        fi;
                rof;
        od;
end;
```

Figure 5: Program Implementing Brélaz's Algorithm

cross the dashed line. The data suggest that $\beta_{128}(.5) = 6$, $\beta_{256}(.5) = 7$, $\beta_{512}(.5) = 8$, and $\beta_{1024}(.5) = 9$. This is consistent with Theorem 2.2, which suggests that $\beta_n(p)$ grows in proportion to $\lambda_n(p)$.

Let k be any integer greater than 2 and let S_k be the set of all graphs with chromatic number no larger than k. The results of this section suggest that most graphs in S_k are easily identified, leading us to the following conjecture.

Conjecture 2.1: For any fixed $k > 0$, the no choice algorithm successfully colors almost all graphs in S_k.

Note that this is not implied by Theorem 2.2, since the probability distribution $X_n(k, p)$ does not assign equal probability to every k-colorable graph.

3. Brélaz's Algorithm

The no choice algorithm is similar to one proposed by Brélaz [2] and justified on experimental grounds. Brélaz's algorithm can be described as a repeated application of the following rule.

> *Coloring Rule 2.* Select an uncolored vertex x that minimizes $|avail(x)|$ and let $c(x) = \min avail(x)$. If there are several vertices available for selection, select one with maximum degree in the uncolored subgraph.

Consider the behavior of Brélaz's algorithm on a normal graph $G \in X_n(k, p)$. The first vertex colored is one of maximum degree, the second is a neighbor of the first which has maximum degree in the uncolored subgraph, the third is a neighbor of the first two which has maximum degree in the uncolored subgraph, and so forth until k vertices have been colored. In other words, during the coloring of the first k vertices, this algorithm mimics the greedy heuristic for clique finding mentioned in the footnote at the end of the previous section. Once the first k vertices have been colored, the algorithm repeatedly selects vertices for which $|avail(x)| = n - k + 1$. That is, it mimics the no choice algorithm. These observations yield the following theorem.

Theorem 3.1: Let $0 < \epsilon < 1$, $0 < p < 1$ be fixed, $k \leq (1 - \epsilon)\lambda(p)$. For almost all $G \in X_n(k, p)$, Brélaz's algorithm produces a k-coloring.

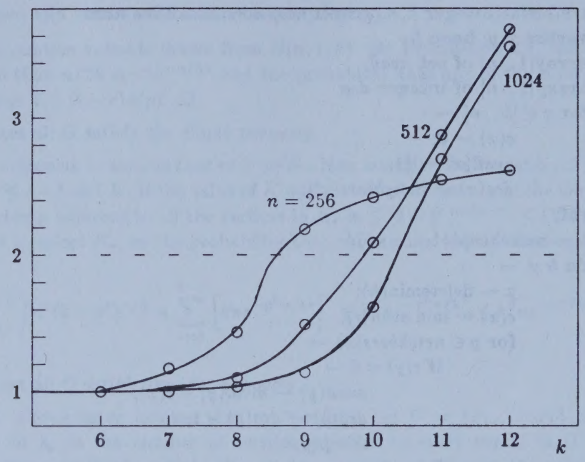


Figure 6: Average Performance Ratio of Brélaz's Algorithm for Graphs in $X_n(k, .5)$

In [2] Brélaz claims an $O(n^2)$ time bound for his algorithm, which is easily proved. In fact, Brélaz's algorithm can be implemented to run in time $O(m \log n)$ for a graph with n vertices and m edges. The program in Figure 5 illustrates this. The heap contains the uncolored vertices. For the purposes of the heap operations, vertex x is smaller than vertex y if $|avail(x)| < |avail(y)|$ or $|avail(x)| = |avail(y)|$ and $deg(x) > deg(y)$. As in the program for the no choice algorithm, the key to an efficient implementation is the data structure used to implement the sets $avail(x)$. If a bit vector is used, the running time is $O(n^2)$. However, using the shrinking set data structure described earlier, each initialization operation can be done constant time, the selection of a minimum can be done in $O(\log n)$ time as can the deletion operation. These observations yield,

Theorem 3.2: Brélaz's algorithm runs in time $O(m \log n)$ on graphs with n vertices and m edges.

Figure 6 shows the results of a series of experiments, which provide more detailed information on the performance of Brélaz's algorithm. One hundred random graphs in $X_n(k, .5)$ were generated for each of several values of n and k, and Brélaz's algorithm was run on each graph. The plot shows the ratio of the average number of colors used to k. As with the no choice algorithm, the performance is quite good for small k, but deteriorates abruptly as k gets large. The point at which the breakdown occurs appears to increase logarithmically with n as one would expect from Theorem 3.2.

4. The Greedy Algorithm

The greedy algorithm for graph coloring is a simple and popular heuristic. It can be described as follows.

For each $x \in [1, n]$, let $c(x) = \min avail(x)$.

Marchetti-Spaccamela and Talamo have shown that for almost all random graphs (in the usual model), the greedy algorithm uses no more than about twice the optimal number of colors. In this section, we study the performance of the greedy algorithm for graphs in $X_n(k, p)$ and conclude that it performs poorly unless k is quite small.

Let $G = (V, E) \in X_n(k, p)$. Let c be the coloring used to generate G and let c' be the coloring computed by the greedy algorithm. We are interested in the probability that c' is a k-coloring. Since almost all G are uniquely k-colorable, this probability is approximately $k!$ times the probability that $c' = c$, for large enough n.

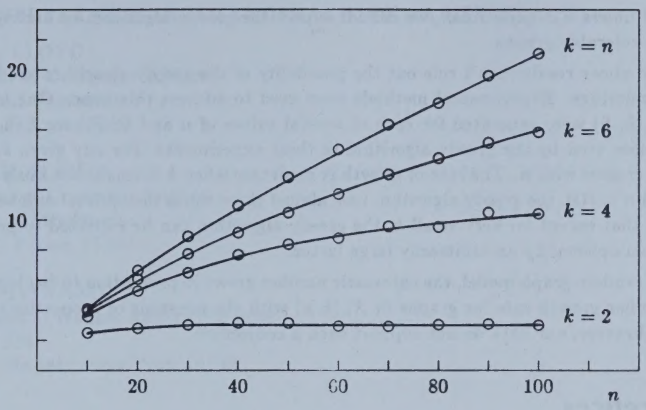


Figure 7: Average Number of Colors Used by Greedy Algorithm for Graphs in $X_n(k, .5)$

Let $S_i(r) = \{1 \le z \le r \mid c(z) = c'(z) = i\}$ for $1 \le i \le k$ and let $P(n_1, n_2, \ldots, n_k)$ be the probability that $|S_i(r)| = n_i$ for all $i \in [1, k]$, where $r = \sum_{i=1}^{k} n_i$. P satisfies the following recurrence.

$$P(0, \ldots, 0) = 1$$
$$P(n_1, \ldots, n_k) = 0 \qquad \text{if any } n_i < 0$$
$$P(n_1, \ldots, n_k) = \frac{1}{k} \sum_{h=1}^{k} P(n_1, \ldots, n_{h-1}, n_h - 1, n_{h+1}, \ldots, n_k) \prod_{j=1}^{h-1} \alpha(n_j) \qquad \text{otherwise}$$

where $\alpha(x) = 1 - (1-p)^x$. (We adopt the convention that an empty product is equal to 1.) Now, let $Q(r)$ be the probability that $c(z) = c'(z)$ for $1 \le z < r$ and $c(r) \ne c'(r)$.

$$Q(r+1) = \sum_{\substack{n_1, \ldots, n_k \ge 0 \\ n_1 + \cdots + n_k = r}} P(n_1, \ldots, n_k) \left[1 - \frac{1}{k} \sum_{h=1}^{k} \prod_{j=1}^{h-1} \alpha(n_j) \right]$$

Now, the probability that $c' \ne c$ is $\sum_{r=1}^{n} Q(r)$. This yields the following theorem.

Theorem 4.1: Let $0 < p < 1$, $k \ge 1$ be fixed and let $G \in X_n(k, p)$. As $n \to \infty$, the probability that the greedy algorithm produces a k-coloring of G approaches $k! \left(1 - \sum_{r=1}^{n} Q(r)\right)$.

The terms in $\sum_{r=1}^{n} Q(r)$ decline rapidly, so for small k, we can use Theorem 4.1 to estimate the probability that the greedy algorithm produces a k-coloring. We illustrate the procedure for the case, $k = 2$. The general equations reduce to

$$P(n_1, n_2) = \frac{1}{2} \left[P(n_1 - 1, n_2) + P(n_1, n_2 - 1) \left(1 - (1-p)^{n_1}\right) \right]$$
$$Q(r+1) = \frac{1}{2} \sum_{n_1=0}^{r} P(n_1, r - n_1)(1-p)^{n_1}$$

Using these equations and Theorem 4.1 we estimate that for large n, the probability of the greedy algorithm successfully 2-coloring a graph in $X_n(2, .5)$ is approximately .42. In the same way, we estimate that the probability of the greedy algorithm successfully 3-coloring a graph in $X_n(3, .5)$ is approximately .091, and the probability of it successfully 4-coloring a graph in $X_n(4, .5)$ is approximately .044. We

conclude that unless k is quite small, we cannot expect the greedy algorithm to find optimal colorings for random k-colorable graphs.

Of course, the above results don't rule out the possibility of the greedy algorithm producing good but sub-optimal colorings. Experimental methods were used to address this issue. One hundred random graphs in $X_n(k, .5)$ were generated for each of several values of n and k. Figure 7 shows the average number of colors used by the greedy algorithm in these experiments. For any given k, the number of colors used increases with n. The rate of growth is moderate when k is small, but fairly large for $k = 6$. For $k = 6$ and $n = 100$, the greedy algorithm uses almost three times the optimal number of colors. The data indicate that except for very small k, the greedy algorithm can be expected to produce colorings that differ from optimal by an arbitrarily large factor.

For the usual random graph model, the chromatic number grows in proportion to $(n/\log n)$. It's natural to guess a similar growth rate for graphs in $X_n(k, p)$ with the constant of proportionality determined by p and k. However, our data do not support such a conjecture.

5. References

1 Angluin, D., L. G. Valiant. "Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings". In *Journal of Computer and System Sciences 18* , 155-193, 1979.

2 Brelaz, Daniel. "New Methods to Color the Vertices of a Graph". In *Communications of the ACM 22*, 251-256, 4/79.

3 Garey, Michael R., David S. Johnson, L. J. Stockmeyer. "Some Simplified NP-complete Graph Problems". In *Theoretical Computer Science 1*, 237-267, 1976.

4 Garey, Michael R., David S. Johnson. "The Complexity of Near-Optimal Graph Coloring". In *Journal of the ACM 23*, 43-49, 1/76.

5 Grimmet, G. R., C. J. H. McDiarmid. "On Colouring Random Graphs". In *Mathematical Proceedings of the Cambridge Philosophical Society 77*, 313-324, 1975.

6 Johnson, David S. "Worst Case Behavior of Graph Coloring Algorithms". In *Proceedings Southeastern Conference on Combinatorics, Graph Theory and Computing*, 513-527, 1974.

7 Karp, Richard M. "Reducibility Among Combinatorial Problems". In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (eds), 1972, Plenum Press.

8 Marchetti-Spaccamela, A., M. Talamo. "Probabilistic Analysis of Graph Colouring Algorithms", Technical report, University of Rome, 1983.

9 Stockmeyer, L. J. "Planar 3-Colorability is NP-complete". In *SIGACT News*, 19-25, 1973.

10 Tarjan, Robert Endre. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, 1983.

11 Wigderson, Avi. "Improving the Performance Guarantee for Approximate Graph Coloring". In *Journal of the ACM*, 729-735, 10/83.

FEEDBACK VERTEX SETS IN POLYNOMIAL TIME - A NEW CLASS*

ERROL L. LLOYD
Department of Computer Science
University of Pittsburgh
Pittsburgh, Penna. 15260

MARY LOU SOFFA
Department of Computer Science
University of Pittsburgh
Pittsburgh, Penna. 15260

CHING-CHY WANG
IBM Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, New York 10598

ABSTRACT

In this paper we present a new class of graphs - the cyclically reducible graphs - for which minimum feedback vertex sets can be found in polynomial time. Unlike existing classes of this type, there is no inclusion relationship between the cyclically reducible graphs and reducible flow graphs. We also define a "transformation-based" class of graphs - the forward reducible graphs - for which minimum feedback vertex sets can be found in time O(e log v). Finally, we show that the two classes which we define are identical, thus providing an O(e log v) algorithm for finding minimum feedback vertex sets of cyclically reducible graphs.

1. OVERVIEW

Since the appearance of the fundamental papers of Cook [Co] and Karp [Ka] on NP-completeness, much work has been done on exploring the consequences of the NP-completeness of various problems. One of the least understood of the original problems discussed in [Ka] is that of finding a **minimum feedback vertex set** of a directed graph G=(V,E). Such a set is defined to be a minimum cardinality subset of V which contains at least one node from every directed cycle in G. These sets arise in numerous applications involving deadlock recovery and so are of practical importance, in addition to being theoretically significant.

Since Karp showed the general problem to be NP-complete, only a few subproblems have been studied. In particular, it has been shown that the problem remains NP-complete for planar graphs with no indegree or outdegree exceeding three [GJ], general graphs with no indegree or outdegree exceeding two [GJ], and edge directed graphs [Ga]. As far as approximation algorithms are concerned, there are no positive results. At present, all of the analyzed methods have a worst case performance which is not within any constant of the optimal [LL1]. The only truly *positive* results on this problem are a linear time algorithm for finding minimum feedback vertex sets in reducible flow graphs [Sh] and several recent results for finding such sets in

* Supported in part by NSF grants MCS-8119341 and MCS-8103713.

specific supersets of the reducible flow graphs [R2,LL2,LSW1].

In this paper we define a new class of graphs - the cyclically reducible graphs - for which minimum feedback vertex sets can be found in polynomial time. This class has an elegant graph theoretic definition and is not restricted to flow graphs. Moreover, even when restricted to flow graphs, the class has no inclusion relationship with the reducible flow graphs. Along with presenting the class and a simple $O(ev^2)$ algorithm for finding minimum feedback vertex sets of graphs in the class, we produce several related results. In particular, we define forward reducible graphs - a "transformation-based" class of graphs - for which minimum feedback vertex sets can be found in time O(e log v). Finally, we show that the cyclically reducible and forward reducible classes are identical, thus providing an O(e log v) method for finding minimum feedback vertex sets in cyclically reducible graphs.

2. CYCLICALLY REDUCIBLE GRAPHS

We begin by introducing the key concepts for defining cyclically reducible graphs. In a directed graph G, a node is **deadlocked** if there is a (possibly trivial) directed path from that node to some directed cycle. The **associated graph of node x with respect to G, A(G,x)**, is the subgraph that contains node x and all nodes of G which are no longer deadlocked if x is removed from G. The edges of A(G,x) are all edges of G which are incident only to nodes in A(G,x). An associated graph is **cyclic** if it contains at least one directed cycle.

Next, we define a **D-sequence** of a digraph G to be a sequence S of nodes $(y_1, ..., y_k)$ such that:

> if the associated graphs of the nodes in S are removed from G (incrementally) in the order in which those nodes appear in S, then each of the associated graphs $A(G_{i-1},y_i)$ is cyclic, where $G_0 = G$ and $G_i = G_{i-1} - A(G_{i-1},y_i)$ for $1 \leqslant i \leqslant k$.

A D-sequence is **complete** if an acyclic graph remains after the removal of the entire sequence of associated graphs. Finally, a directed graph G is **cyclically reducible** if and only if there exists a complete D-sequence for G. This definition leads to the following correspondence:

Theorem 1: If $(y_1, ..., y_k)$ is a complete D-sequence of G, then $\{y_1, ..., y_k\}$ is a minimum feedback vertex set of G.

Proof: Assume that $(y_1, ..., y_k)$ is a complete D-sequence of digraph $G=(V,E)$. It follows from the definition of complete D-sequences, that $\{y_1, ..., y_k\}$ is indeed a feedback vertex set of G (in particular, G_k is acyclic). Thus, we need only show that $\{y_1, ..., y_k\}$ is a *minimum* feedback vertex set.

Hence, for each i, $1 \leqslant i \leqslant k$, let L_i be a cycle in $A(G_{i-1}, y_i)$. Clearly, every feedback vertex set of G must contain at least one node from each of these cycles. Moreover, since

these cycles are pairwise node disjoint, every feedback vertex set of G contains at least k nodes. Hence, $\{y_1, ..., y_k\}$ is a minimum feedback vertex set of G. ■

The above theorem and definitions suggest certain approaches to finding a minimum feedback vertex set of G. In particular, one might attempt to find a minimum feedback vertex set of a cyclically reducible graph G by locating a complete D-sequence of G. One method for finding such a complete D-sequence is:

> Locate a node x in G such that A(G,x) is cyclic and remove A(G,x) from G. Repeat the process on the now reduced G. If there is more than one choice for x, then select x arbitrarily (or perhaps by some more specific criterion).

If the process completes, then the nodes whose associated graphs were removed form a complete D-sequence of G, and hence, by Theorem 1, a minimum feedback vertex set of G. What happens however, if the process does not complete? That is, a point is reached where no remaining node has a cyclic associated graph. The only alternative appears to be "backing up" to the nearest point at which there was a choice of x's with cyclic associated graphs, and trying another alternative for x. Unfortunately, because of the backtracking, the process may require exponential time in the worst case. The ideal, if possible, is to select from the alternatives for x in such a manner that we never have to backtrack. Somewhat surprisingly, the following theorem shows that such a selection among alternatives for x can be done arbitrarily. A proof of the theorem appears in [WLS].

Theorem 2: If G is a cyclically reducible graph having a node x such that A(G,x) is cyclic, then there is a complete D-sequence for G having x as its first node.

This result shows that the definition of cyclically reducible graphs satisfies a kind of "Church-Rosser" property [R1] with respect to complete D-sequences. With this property, the definition of a cyclically reducible graph is useful from an algorithmic point of view as it establishes that the algorithm mentioned above (with the backtracking eliminated) can be used to find a minimum feedback vertex set of a cyclically reducible graph by finding a complete D-sequence of the graph. A straight-forward implementation of the algorithm yields an $O(ev^2)$ running time.

3. A TRANSFORMATION-BASED CLASS

In this section we describe another new class of graphs. This class is based on the following simple and easily recognizable graph transformations:

Self loop-transformation - Given a node x with a self-loop, remove x and all edges incident to x from G.

Outdegree 0-transformation - Given a node x with outdegree 0, remove x and all edges incident to x from G.

Outdegree 1-transformation - Given a node x with outdegree 1 and no self-loop, combine x and its only successor, y, into a single node.

Given the above transformations, a **F-sequence** for G is a sequence of pairs of transformations and nodes such that each transformation can be applied to its corresponding node when the transformations are applied successively in the order in which they appear in the sequence. A F-sequence is **complete** if the graph is empty after applying the entire sequence. A graph is **forward reducible** if there exists a complete F-sequence for that graph.

Similarly to the cyclically reducible graphs, we now provide a relationship between complete F-sequences and feedback vertex sets:

Theorem 3: If G is a forward reducible graph and τ is a complete F-sequence for G, then $\{y_1, ..., y_k\}$ is a minimum feedback vertex set of G, where $y_1, ..., y_k$ are the nodes of G to which the self-loop transformation is applied in τ.

Proof: Inductively, we assume that the theorem holds for all graphs with fewer nodes than G. We begin by considering $<T,x>$, the first element of τ. We let τ' be identical to τ with that first element omitted, and let graph $G' = T(G,x)$. Obviously, τ' is a complete F-sequence for G', and by the induction hypothesis, $Y = \{y_1, ..., y_k\}$ is a minimum feedback vertex set of G', where $y_1, ..., y_k$ are the nodes of G' to which the loop-transformation is applied in τ'. Note that in G, the loop-transformation is also applied to each of $y_1, ..., y_k$. There are three possibilities for T (the first transformation in τ).

Case 1: T is an outdegree 0-transformation

Y is also a minimum feedback vertex set of G, since x does not lie on any cycle in G.

Case 2: T is the loop-transformation

Since removing x from G breaks no cycles in G', and the removal of Y from G cannot break the self-loop at x, it follows that $Y \cup \{x\}$ is a minimum feedback vertex set of G.

Case 3: T is an outdegree 1-transformation

We claim that Y is also a minimum feedback vertex set of G. Note that establishing that Y is a feedback vertex set of G is enough, since the minimality will follow from the minimality of Y with respect to G'. Thus, consider any cycle $C = (z_1, ..., z_s)$ in G. If x is not on C, then C is also present in G', and is broken by the removal of Y. If x is on C, say $z_i = x$, then, it follows from the definitions of the outdegree 1 transformation, that $(z_1, ..., z_{i-1}, z_{i+1}, ..., z_s)$ is a cycle in G'. But then some node z_j from that cycle is in Y. The removal of Y (in particular, z_j) from G breaks cycle C. ■

Next we show that a Church-Rosser property [R1] holds with respect to complete F-sequences:

Theorem 4: If G is a forward reducible graph and transformation T is applicable to x in G, then there is a complete F-sequence of G having $<T,x>$ as its first element.

Proof: Inductively, we assume that the theorem holds for all graphs with fewer nodes than G. Since G is a forward reducible graph, there exists a complete F-sequence τ for G. Let $<T_s,y>$ be the first element of τ and consider $G' = T_s(G,y)$. There must exist a transformation T_k which is applicable to x in G' for no transformation applied to y can increase the outdegree of x or remove a self loop on x. Likewise, there exists a transformation T_m which is applicable to y in $T_i(G,x)$. By way of induction, we assume that $<T_k,x>$ is the second element of τ. Now consider a T-sequence τ' which is identical to τ, except that $<T_i,x>$ and $<T_m,y>$ are substituted for $<T_s,y>$ and $<T_k,x>$ as the first two elements of τ.

We now want to show that $T_k(T_s(G,y),x) = T_m(T_i(G,x),y)$. We note that the two graphs in question have the same node set, so without loss of generality, assume that edge (z,w) is present in $T_k(T_s(G,y),x)$ and is not present in $T_m(T_i(G,x),y)$. It follows from the definitions of the transformations that (z,w) is not present in G and is created in the process of transforming G into $T_k(T_s(G,y),x)$. Since neither the loop-transformation nor the outdegree 0-transformation creates edges, an outdegree 1-transformation must be applied in creating (z,w). There are two cases to consider.

Case 1: (z,y) and (y,w) are in G and T_s is an outdegree 1-transformation

Since edges (z,y) and (y,w) are present in G, they must also be present in $T_i(G,x)$. Then, since edge (z,w) is not present in $T_m(T_i(G,x),y)$, it must be that T_m is the loop-transformation. This means that the application of T_i to x in G creates the self-loop on y. It follows that edges (z,y), (y,w), (y,x) and (x,y) are all present in G. Thus, in G, the outdegree of y is at least two - contradicting the fact that T_s is an outdegree 1-transformation.

Case 2: (z,x) and (x,w) are in $T_s(G,y)$ and T_k is an outdegree 1-transformation

We begin by considering T_i. Since (z,x) and (x,w) are present in $T_s(G,y)$, it follows that T_i is not an outdegree 0-transformation. If T_i is the loop-transformation, then T_k would also have to be the loop-transformation, since the self-loop on x could not be affected by any transformation applied to y. Thus, T_i is an outdegree 1-transformation. It follows that edges (z,x) and (x,w) are not both present in G, since if they were, then edge (z,w)

would be present in $T_i(G,x)$ and hence, in $T_m(T_i(G,x),y)$. Thus, we assume without loss of generality that (z,x) is not present in G. Since (z,x) is present in $T_s(G,y)$, it must be that T_s is an outdegree 1-transformation and that edges (z,y) and (y,x) are present in G. Moreover, either edge (x,w) or edges (x,y) and (y,w) are present in G. The latter situation is handled by case 1, so suppose that edge (x,w) is present in G. Since T_i is an outdegree 1-transformation, edge (y,w) is present in $T_i(G,x)$. Since (z,y) is present in G, it is also present in $T_i(G,x)$, and since (z,w) is not in $T_m(T_i(G,x),y)$, T_m cannot be an outdegree 1-transformation. Thus, T_m is the loop-transformation. Since T_s is an outdegree 1-transformation, it must be that the application of T_i to x in G creates the self-loop on y. It follows that edges (y,x) and (x,y) are present in G. Finally, since T_s is an outdegree 1-transformation, there is a self-loop on x in $T_s(G,y)$. Thus, T_k is the loop-transformation - a contradiction.

Thus, $T_k(T_s(G,y),x) = T_m(T_i(G,x),y)$. Since τ is a complete T-sequence for G, it follows that τ' is also a complete T-sequence for G. ■

Based on Theorems 3 and 4, we have the following algorithm for (simultaneously) recognizing forward reducible graphs and finding minimum feedback vertex sets for such graphs:

> Given G, choose a transformation T and a node x, such that T can be applied to x; update G by applying that transformation; if T is the loop-transformation, then place x into the feedback vertex set; repeat the process on the now transformed G. The algorithm completes when G is empty or when no transformation can be applied.

It follows from Theorems 3 and 4, that the algorithm completes with G empty if and only if G is a forward reducible graph, and in this case, the feedback vertex set is of minimum cardinality. An O(e log v) implementation of the above algorithm is given in [LSW1]. An alternative O(e log v) algorithm follows from the work of [LL2].

Finally, we conclude the paper by proving that the two classes of graphs we have defined are, in fact, identical:

Theorem 5: G is cyclically reducible if and only if G is forward reducible.

Proof: We begin by assuming that G is forward reducible and prove that G is cyclically reducible. The proof is by induction on the number of nodes in G. Thus, we consider G, a forward reducible graph, and assume that the theorem holds for all forward reducible graphs having fewer nodes than G. In particular, if transformation T is applicable to x in G, then graph $G' = T(G,x)$ is forward reducible, and hence is cyclically reducible. Thus, let $(y_1, ..., y_k)$ be a complete D-sequence for G'. We claim that $(y_1, ..., y_k)$ is also a complete D-sequence for G provided T is not the loop-transformation. If T is the loop-transformation, then we claim that (x,

y_1, ..., y_k) is a complete D-sequence for G. These claims can be substantiated by considering the three possible transformations for T. The details of the proofs are in [LSW2].

We next show that if G is cyclically reducible, then G is forward reducible. Inductively, we assume that the result holds for all cyclically reducible graphs with fewer nodes than G. Let S be a complete D-sequence for G and let x be the first node in S. Consider the associated graph A(G,x). If that graph consists only of x, then x has a self-loop in G. We let G′ denote T(G,x), where T is the loop-transformation. Then G′ is a cyclically reducible graph, which has fewer nodes than G, and by the induction hypothesis, G′ is forward reducible. We let S′ be a complete F-sequence for G′. Then, by appending <T,x> as the first transformation of that sequence, we have a complete F-sequence for G, and therefore G is forward reducible.

Next we assume that A(G,x) contains at least one node other than x. Let z be such a node which has outdegree zero in G - {x}. If z also has outdegree zero in G, then the result inductively holds in a fashion similar to that above, except that the outdegree 0-transformation is substituted for the loop-transformation. Thus, we assume that z has outdegree one in G, and consider G′ = T(G,z), where T is not the outdegree 1-transformation. We claim that S is also a complete D-sequence for G′. Provided this is the case, then G′ is cyclically reducible, and the result follows as above.

To see that S is a complete D-sequence for G′, it is enough to show that G - A(G,x) = G′ - A(G′,x). Recall that G - A(G,x) consists of all nodes which are deadlocked in G - {x} and that G′ - A(G′,x) consists of all nodes which are deadlocked in G′ - {x}.

Case 1: w is deadlocked in G - {x}

Thus, there exists a path from w to some directed cycle in G - {x}. Since z is in A(G,x), z cannot be on that path or cycle. It follows that the path and cycle are also present in G′ - {x} (note that all of the edges modified in forming G′ from G involve z or x, and the edges in the path and cycle involve neither). Thus, w is also deadlocked in G′.

Case 2: w is deadlocked in G′ - {x}

Thus, there exists a path from w to some directed cycle in G′ - {x}. Note that neither x nor z can be on this cycle, since neither is present in G′ - {x}. Again, each of the edges in the path and cycle must be present in G - {x}, and therefore w is deadlocked there. ■

4. RELATED RESULTS

In related work [LSW2], the class of cyclically reducible graphs are included as part of a hierarchy of classes of graphs for which minimum feedback vertex sets can be found in polynomial time. Besides the cyclically reducible graphs, the hierarchy includes the reducible flow graphs and an infinite sequence of classes based on an algorithm of Smith and Walford [SW] for finding minimum feedback vertex sets in arbitrary graphs. In addition, two other

transformation-based classes of graphs are also included in the hierarchy. Both cyclically reducible graphs and reducible flow graphs are shown to be properly included in the Smith/Walford class as well as in a transformation-based class consisting of five transformations.

Acknowledgements: R.E.Tarjan and B.K. Rosen provided useful suggestions and comments on various portions of this work.

5. REFERENCES

[Co] Cook, S.A., "The complexity of theorem-proving procedures," Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 1971, 151-158.

[Ga] Gavril, F., "Some NP-compete problems on graphs," Proceedings of the 11th Conference on Information Science and Systems, Johns Hopkins Univ., 1977, 91-95.

[GJ] Garey, M.R. and Johnson, D.S., *Computers and Intractability - A Guide to the Theory of NP-Completeness,* W.H. Freeman and Company, 1979.

[Ka] Karp, R.M., "Reducibility among combinatorial problems", in R.E. Miller and J.W. Thatcher (eds.), *Complexity of Computer Computations,* Plenum Press, New York, pp. 85-103.

[LL1] Leung, J.Y.T., and Lai, E.K., "On minimum cost recovery from system deadlock", IEEE Trans. on Computers, 28(9) 1979, 671-677.

[LL2] Levy, H. and Low, D.W., "A new algorithm for finding small cycle cutsets," IBM Los Angeles Scientific Center Report G320-2721, 1983.

[LSW1] Lloyd, E.L., Soffa, M.L., and Wang, C.C., "On locating minimum feedback vertex sets in polynomial time," To appear as a University of Pittsburgh, Department of Computer Science Technical Report.

[LSW2] Lloyd, E.L., Soffa, M.L., and Wang, C.C., "Minimum feedback vertex sets - a graph hierarchy", in preparation.

[R1] Rosen, Barry K., "Tree-manipulating systems and Church-Rosser theorems," JACM 20(1) 1973, 160-187.

[R2] Rosen, Barry K., "Robust Linear Algorithms for Cutsets", Journal of Algorithms, 3 1982, 205-217.

[Sh] Shamir, A., "A linear time algorithm for finding minimum cutsets in reducible graphs", SIAM J. Computing, Vol. 8 No. 4 (1979), pp. 645-655.

[SW] Smith, G.W. and Walford, R.B., "The identification of a minimal feedback vertex set of a directed graph," IEEE Trans. on Circuits and Systems, Vol. CAS-22 No. 1 (1975), pp. 9-14.

[WLS] Wang, C.C., Lloyd, E.L., Soffa, M.L., "Feedback vertex sets and cyclically reducible graphs", submitted for publication.

ROBUST ESTIMATION OF AR MODELS USING THE LATTICE STRUCTURE

SHIPING LI and BRADLEY W. DICKINSON
Dept. of Electrical Engineering & Computer Science
Princeton University
Princeton, NJ 08544

ABSTRACT

A new approach to the robust estimation of autoregressive models (with prespecified order) for time series with additive outliers is introduced. The approach is inspired by the idea of data cleaning by filtering or interpolation proposed by R. D. Martin and D. J. Thomson in their work on robust spectrum estimation. The main feature of the new approach is that the lattice structure is used as the parametric model for the AR process in the parameter estimation and data cleaning process. Therefore both the forward and backward prediction errors can be used in cleaning the data. This will greatly increase the accuracy of cutting out bad data, making the parameter estimates more resistant to contamination by "outliers".

I. INTRODUCTION

Linear prediction and autoregressive modeling have been much studied in statistics, econometrics and engineering applications. Accurately determining the order of an AR model and estimating the model parameters are very important in these studies. In this paper we will focus our attention on some aspects of the parameter estimation problem. Specifically, we will be concerned with how to estimate the AR parameters robustly, assuming that the order of the AR model is known.

The least squares method has dominated the AR model estimation problem because it performs well for the ideal case, we have the model

$$x(n) = -\sum_{i=1}^{p} a_i x(n-i) + e(n) \tag{1}$$

$$y(n) = x(n) + v(n) \tag{2}$$

where $\{e(n)\}$ is the white Gaussian innovation process, and $v(n)=0$, that is, assuming no additive noise. In this case the estimation problem based on the observations $\{y(n)\}$ requires solving the so-called covariance equations, the solutions of the covariance equations give good estimates of the AR parameters.

In reality the observations of an AR process are always contaminated by additive noise and/or observation errors, i.e., $v(n)\neq 0$. In such cases the estimates given by the least squares method are very poor. Efforts have been made by some researchers to solve this problem, that is, to make the parameter estimation schemes more robust (see, for example, [4], [5], [6]). Here we will present a new approach to the robust estimation of autoregressive models (with prespecified order) for time series with additive outliers. Our approach is inspired by the idea proposed by R. D. Martin and D. J. Thomson [5] in their work on robust spectrum estimation. In [5], Martin and Thomson first did "data cleaning" by filtering or interpolation, then used the cleaned data to do the spectrum estimation.

The main feature of the new approach is that the lattice structure [2] is used as the parametric model for the AR process in the parameter estimation and data cleaning process. Therefore both the forward prediction errors and backward prediction errors can be used in cleaning the data. This will greatly increase the accuracy of weeding out outliers from the contaminated data.

II. DATA CLEANING WITH THE LATTICE STRUCTURE

The lattice structure is known to have many advantageous properties, Here we will show another nice property particular to the lattice structure; namely, it is more suitable for applying the data cleaning approach to estimating the AR model parameters robustly. We will outline the approach, using the example of a third order stationary AR process with additive noise described by

$$x(n) = -a_1x(n-1) - a_2x(n-2) - a_3x(n-3) + e(n) \tag{3}$$

$$y(n) = x(n)+v(n) \tag{4}$$

where $\{ e(n) \}$ is as in (1), and $\{ v(n) \}$ is the additive noise (the contamination). Here, we assume that $\{ v(n) \}$ is an i.i.d. sequence drawn from the distribution

$$f(v) = (1-\varepsilon)\delta(v)+\varepsilon N(0,\sigma_v^2) \tag{5}$$

That is, we are assuming that most data are observed accurately, with only a small fraction subject to additive errors which obey the Normal distribution with appropriate variance σ_v^2. Actually, it is not necessary to assume the Normal distribution for $v(n)$ in our approach, which does not depend on the distribution of $v(n)$.

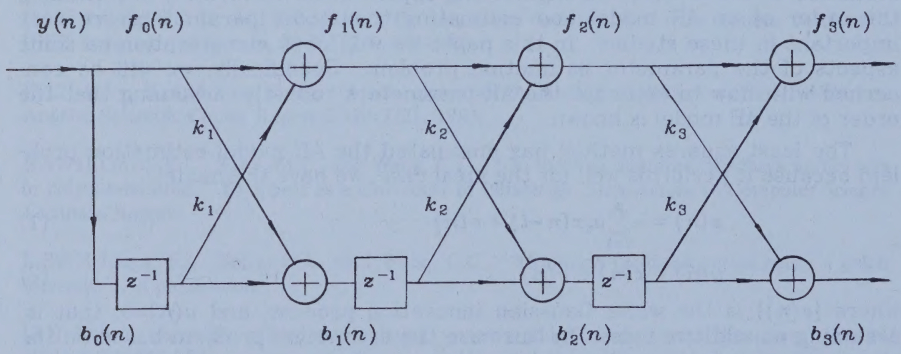


Figure 1. Third-order Lattice Structure

Using a lattice estimation algorithm, a set of PARtial CORrelation coefficients (PARCOR) $\{ \hat{k}_1, \hat{k}_1, \hat{k}_3 \}$ of a third order prediction error filter are estimated with the contaminated data $\{ y(n) \}$. One way of computing the PARCOR is the Burg algorithm [1],

$$\hat{k}_{i+1} = -\frac{2\sum_{n=i+1}^{N} f_i(n)b_i(n-1)}{\sum_{i+1}^{N}[f_i(n)^2+b_i(n-1)^2]} \tag{6}$$

$$f_{i+1}(n) = f_i(n) + \hat{k}_{i+1}b_i(n-1) \tag{7}$$

$$b_{i+1}(n) = b_i(n-1) + \hat{k}_{i+1}f_i(n) \qquad i=1, 2, 3 \tag{8}$$

with $f_0(n)=b_0(n)=y(n)$. From the PARCOR estimates $\{\hat{k}_1, \hat{k}_2, \hat{k}_3\}$, we can easily get the corresponding parameters $\{ \hat{a}_1, \hat{a}_2, \hat{a}_3 \}$, which are also called the Linear Prediction Coefficients (LPC).

Although both the lattice form filter and the recursive filter corresponding to these two sets of parameters (PARCOR and LPC) will be far from accurate estimates of the desired model, by passing the observed data $\{y(n)\}$ through the lattice whitening filter, we can get the forward and backward prediction errors $\{ f_3(n), b_3(n) \}$,

$$f_3(n) = y(n) + \hat{a}_1 y(n-1) + \hat{a}_2 y(n-2) + \hat{a}_3 y(n-3) \tag{9}$$

$$b_3(n) = y(n-3) + \hat{a}_3 y(n-2) + \hat{a}_2 y(n-1) + \hat{a}_1 y(n) \tag{10}$$

Together, they reveal quite accurately the positions and the amplitudes of the outliers. Then we "clean" the data by replacing those outliers with some appropriate values. Various different schemes have been tried for doing the cleaning job. What followed is a brief of description some simple data cleaning schemes we have considered.

A). Identifying which data contain outliers

The first step in cleaning the data is to determine which data points contains outliers. Suppose that $y(n)$ contains an outlier for some n. From equations (9), (10), it is fairly easy to see that $f_3(n)$, $b_3(n)$ will most likely be relatively large, similarly $f_3(n+1)$, $f_3(n+2)$, $f_3(n+3)$ and $b_3(n+1)$, $b_3(n+2)$, $b_3(n+3)$ will also be relatively large. Our computer simulations have confirmed this and shown that this is especially true for $f_3(n)$, $f_3(n+1)$ and $b_3(n+3)$, $b_3(n+2)$.

A simple scheme which distinguishes the bad data from the observations is to monitor the errors $\{f_3(i)\}$ and $\{b_3(i)\}$. If both $f_3(i)$ and $b_3(i+3)$ are large for some i, then we decide that $y(i)$ contains an outlier. Otherwise we decide that $y(i)$ is a good observation.

Not unexpectedly, the signs of these $f_3(\cdot)$ and $b_3(\cdot)$ corresponding to an outlier at position n also have some interesting patterns. If $f_3(n)$ is positive, then $f_3(n+1)$ will be negative. Similarly $b_3(n+3)$ and $b_3(n+2)$ will also have different signs. And, $f_3(n)$ and $b_3(n+3)$ will have same sign. These relations are not difficult to see by analogy with linear regression. So we can also monitor the signs and magnitudes of $\{ -f_3(i)f_3(i+1),\ -b_3(i+3)b_3(i+2) \}$. For those terms corresponding to outliers the signs will be positive and the magnitudes will be relatively large. In this way we can increase the accuracy of singling out the outliers.

More general schemes, which also make use of $f_3(i+2)$, $b_3(i+1)$, etc., will do better jobs but at the expense of increased computation.

In the above schemes, we need some values as references for comparison with the residuals $\{f_3(n)\}$ and $\{b_3(n)\}$ so we can say whether or not an observation has been contaminated. This is the scale estimation problem. In our problem, the scales should also be estimated robustly, the so-called Median Absolute Deviation (MAD) [3] can be used as the robust estimates of scales.

B). Replace those data singled out in step A).

The second step of data cleaning is to replace those bad data singled out as described above. One reasonable way of doing this is to replace $y(i)$ by, say, the prediction $\hat{y}_f(i)=y(i)-f_3(i)$ or $\hat{y}_b(i)=y(i)-b_3(i+3)$, or some weighted average of $\hat{y}_f(i)$ and $\hat{y}_b(i)$. A more general form of the replacement we propose to use can be written as

$$y(i) \leftarrow (1-\alpha_i)[k_f f_3(i)+k_b b_3(i+3)] \tag{11}$$

where k_f and k_b are weighting coefficients with $k_f+k_b=1$, and $\alpha_i=\alpha(f_3(i), b_3(i+3))$ is a function of $f_3(i)$ and $b_3(i+3)$.

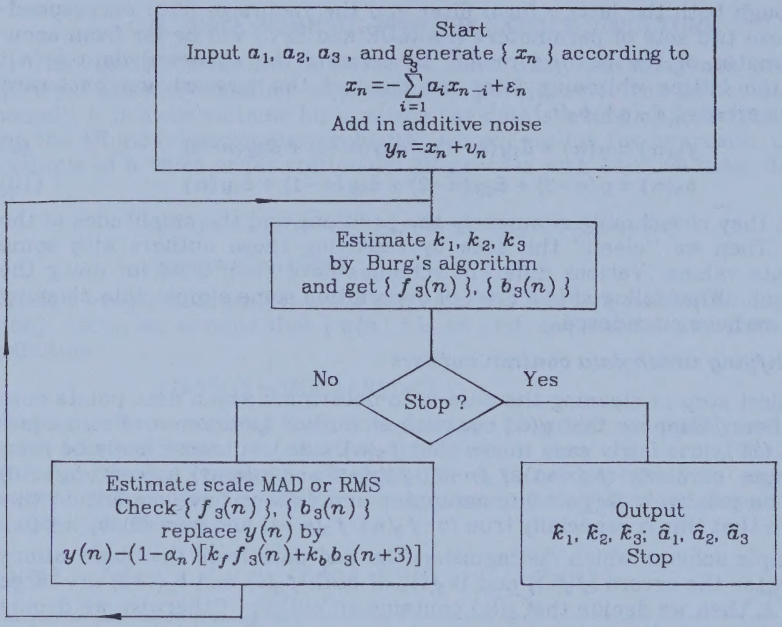


Figure 3. The flowchart of the simulations

IV. DISCUSSIONS

In the data cleaning approach to the robust estimation of time series models, it is easier to use filtering than to use interpolation. Obviously using interpolation would give a much better result, because it uses more information in cleaning the data. But it is much difficult to implement.

As we said earlier, the advantage of using the lattice structure in the data cleaning procedure is that there are two prediction errors available, the forward prediction error resulted from predicting the present datum using the past observations, and the backward prediction error resulted from predicting the present datum using the future observations. This is close in spirit to interpolation, which uses both past and future observations to get a smoother estimate. But with the lattice it is much easier to realize, and the computation amount is about the same as that of filtering.

Our approach works quite well when the percentage of bad data is not very high, as indicated by simulations. One problem with this approach is that it sometimes fails to give good results when two or more outliers appear, separated by less than the model order. As the percentage of outliers increase, there will be more chance for two or more outliers to appear in a small neighborhood, so the performance can be expected to decline gradually.

The shortcoming of using the MAD as the scale estimate is that it involves two passes by a median operation. One reasonable substitute is to use the root mean squared value (RMS), $\hat{\sigma}_f = \sqrt{\frac{1}{N}\sum_i f_3^2(n)}$ and $\hat{\sigma}_b = \sqrt{\frac{1}{N}\sum_i b_3^2(n)}$, which will reduce the computation amount considerably. Of course the performance will be somewhat inferior to that of MAD because RMS is not a robust estimate of the scale.

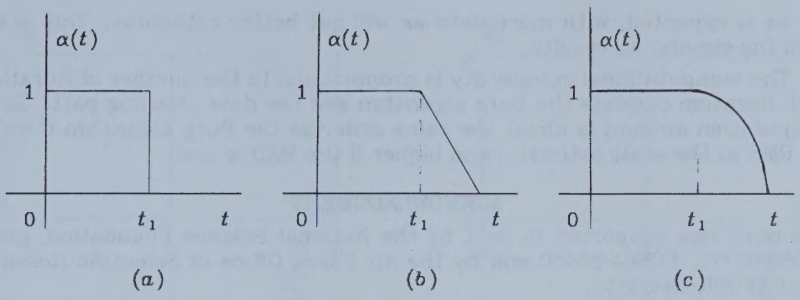


Figure-2. three α-functions

In Figure 2 we have plotted several possible α-*functions*, where t is related to $f_3(i)$ and $b_3(i+3)$ by

$$t : \{|f_3(i)| < t\} \cup \{|b_3(i+3)| < t\} \tag{12}$$

The α-functions here can be regarded as some sort of weighting functions or windows. They let those data with small residues pass and single out those with larger residues.

After the data cleaning stage, we can estimate a second set of the PARCOR and the LPC parameters with the Burg algorithm using the cleaned data. These operations can be repeated several times, until some criterion is satisfied. Our computer simulations showed that in most cases we can get reasonably good results by just one iteration. Further iteration will give some more improvement, but not much.

III. COMPUTER SIMULATIONS

According to the approach outlined above, we did some computer simulations. The flowchart for the simulations is shown in Figure 3. In Figure 4 to Figure 7 we give some simulation results, which are obtained with 2% of outliers, the variance σ_v^2 of the outliers is two times of the mean squared value of the AR series; the lengths of observations are N=100, and N=400; the actual parameters are k_1=0.8722, k_2=−0.4967, k_3=−0.6769 and a_1=−0.9692, a_2=−0.3869, a_3=0.6769, the corresponding poles are p_1=0.95, $p_2=0.75e^{j0.86\pi}$, and $p_3=0.75e^{-j0.86\pi}$ in the *z−plane*. The scatter plots use the LPC estimates $\hat{a}_1$, $\hat{a}_2$, $\hat{a}_3$ or the PARCOR estimates $\hat{k}_1$, $\hat{k}_2$, $\hat{k}_3$ as the coordinates. There are fifty points in each scatter plot, representing fifty repetitions of the same simulation, so we can roughly see something about the statistics of the estimates.

Simulations with parameters different from those given above have also been done. Specifically, we used 2%, 3%, 5%, 10% outliers with different variances, and data length of N=100, N=200, N=400, N=800, N=1600, and AR parameters with different poles locations inside the unit circle in the *z−plane*. All simulation results showed significant improvement to the robustness of the Burg algorithm in estimating the AR parameters after the data cleaning operation has been performed.

From the scatter plots we see that when no outliers are added, the Burg algorithm gave pretty good estimates of the parameters of AR models. But when a small fraction outliers are added, the estimates drifted away from the true values and spread out significantly. After performing the data cleaning operations for just one iteration, we again got estimates very close to those obtained when no outliers were added.

As is expected, with more data we will get better estimates. This is seen from the simulation results.

The computational complexity is proportional to the number of iterations. Each iteration consists the Burg algorithm and the data cleaning part. So the computation amount is about the same order as the Burg algorithm if we use the RMS as the scale estimate, and higher if the MAD is used.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation, grants ECS82-05772, ECS84-05460 and by the Air Force Office of Scientific Research, grant AFOSR-84-0381.

REFERENCES

[1] J. P. Burg, "Maximum Entropy Spectral Analysis," presented at the 37th Annu. Meeting Soc. of Exploration Geophysicists, Oklahoma City, Okla., Oct. 31, 1967.

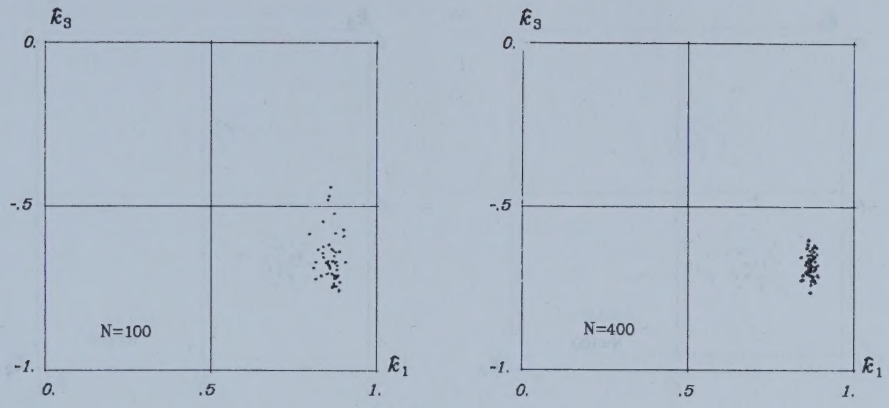
[2] F. Itakura and S. Saito, "Digital Filtering Techniques for Speech Analysis and Synthesis," in *Proc. 7th. Int. Conf. on Accoustics*, Budapest, 1971, Paper 25-c-1.

[3] P. J. Huber, *Robust Statistics*, John Wiley & Sons, 1981.

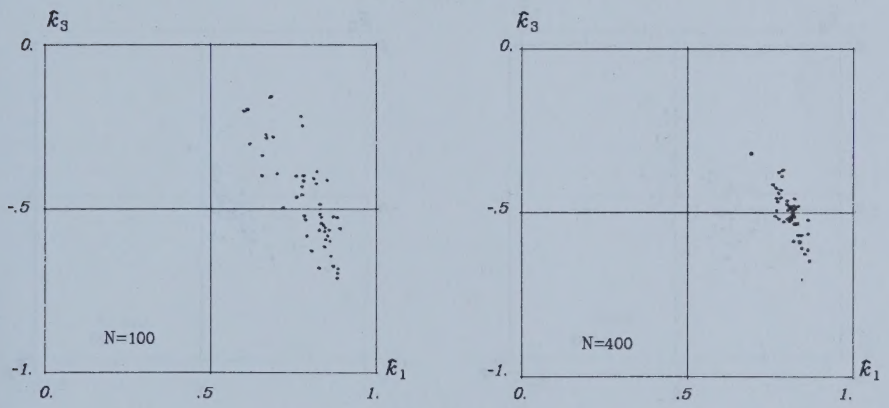
[4] R. D. Martin, "Robust Estimation of Autoregression Models," in *Directions in Time Series*, edited by Brillinger, et al., Inst. of Math. Statistics Publication.

[5] R. D. Martin and D. J. Thomson, "Robust-Resistant Spectrum Estimation," *Proc. IEEE*, Vol. 70, No. 9, Sept. 1982, pp.1097-1115.

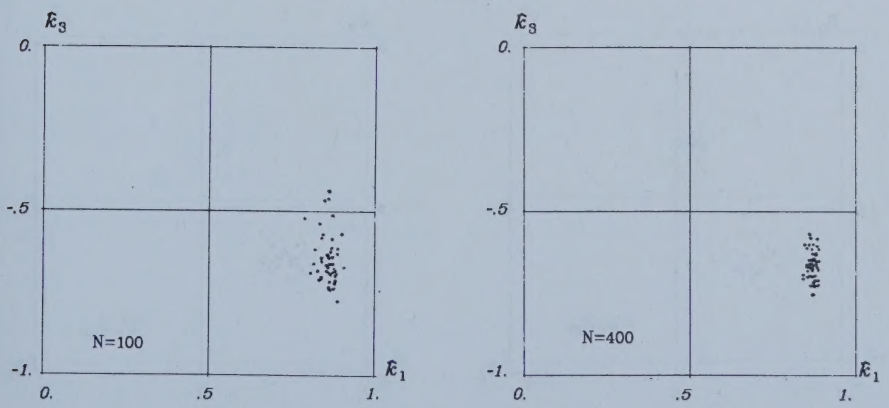
[6] R. D. Martin, "Robust Methods for Time Series," in *Applied Time Series II*, D. F. Findley, Ed. New York: Academic Press, 1981



(a) Without outliers

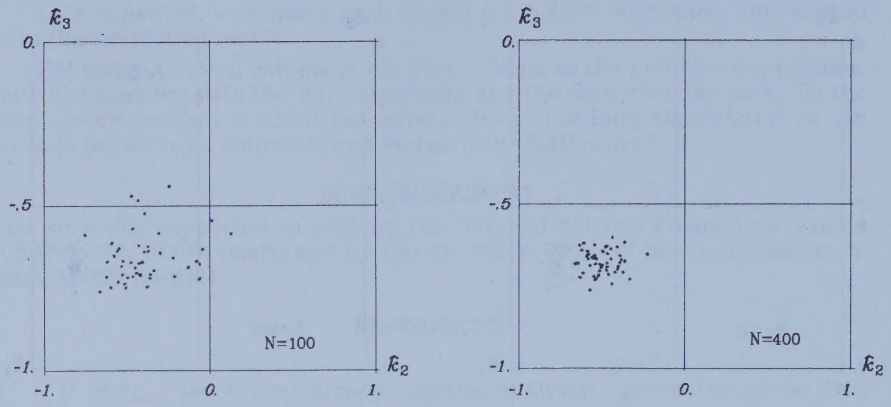


(b) With outliers

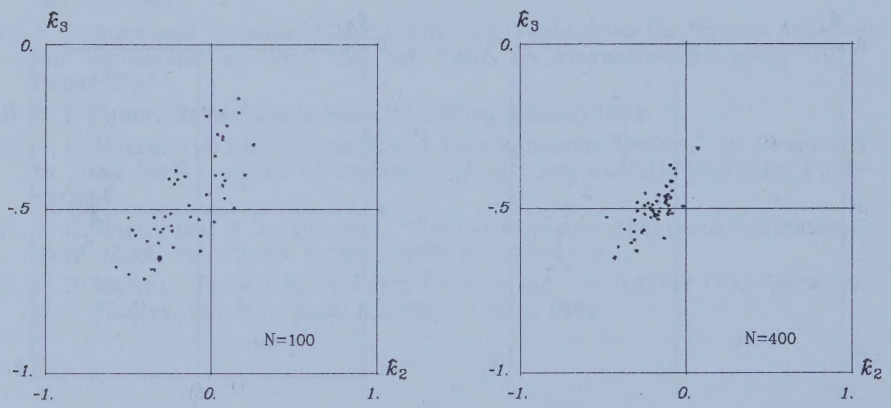


(c) With data cleaning

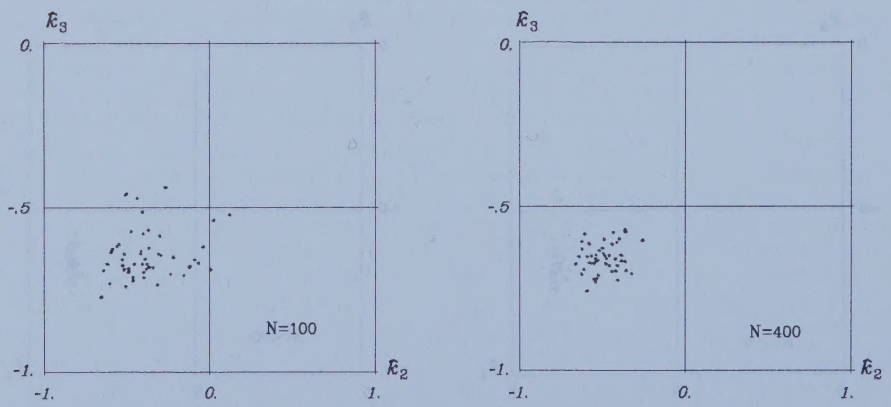
Figure 4. Estimates of PARCOR $\hat{k}_3$ v.s. $\hat{k}_1$ by the Burg algorithm in three cases: without outliers, with outliers, and after data cleaning



(a) Without outliers

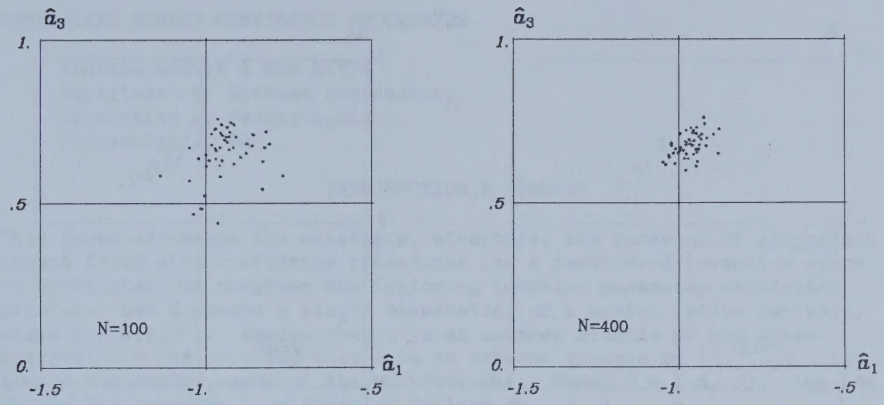


(b) With outliers

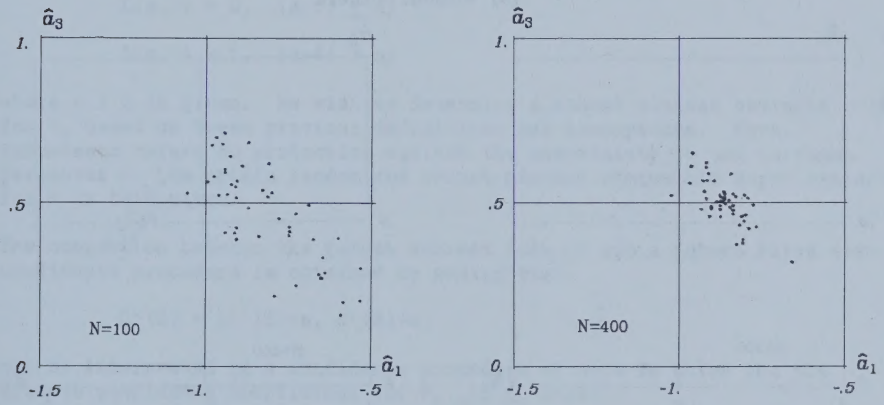


(c) With data cleaning

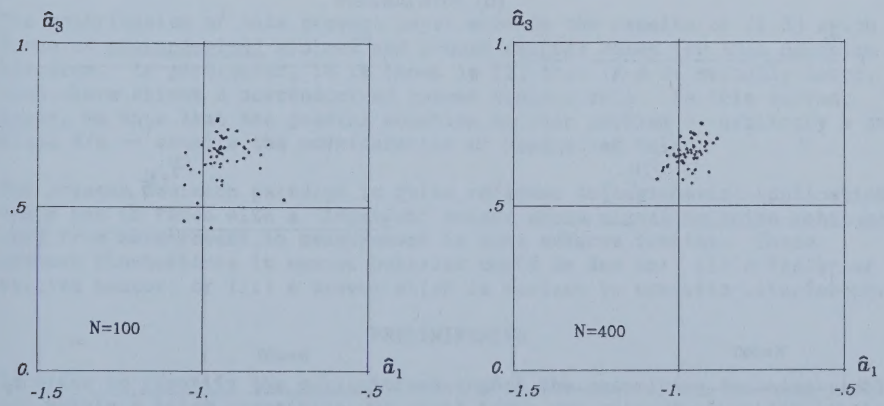
Figure 5. Estimates of PARCOR $\hat{k}_3$ v.s. $\hat{k}_2$ by the Burg algorithm in three cases: without outliers, with outliers, and after data cleaning



(a) Without outliers

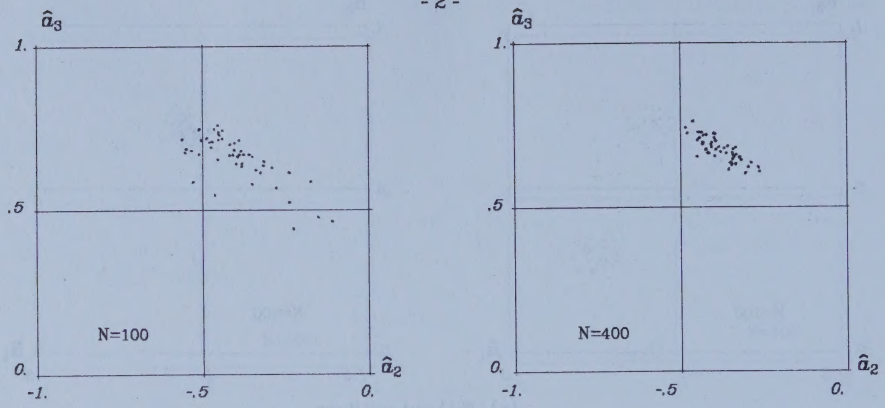


(b) with outliers

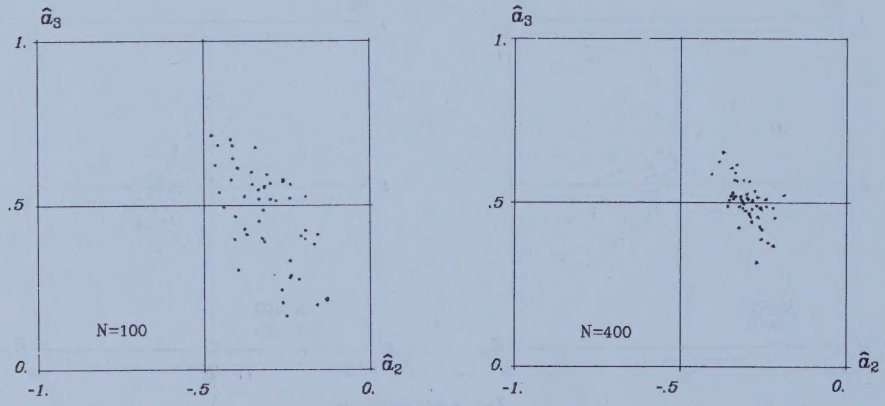


(c) With data cleaning

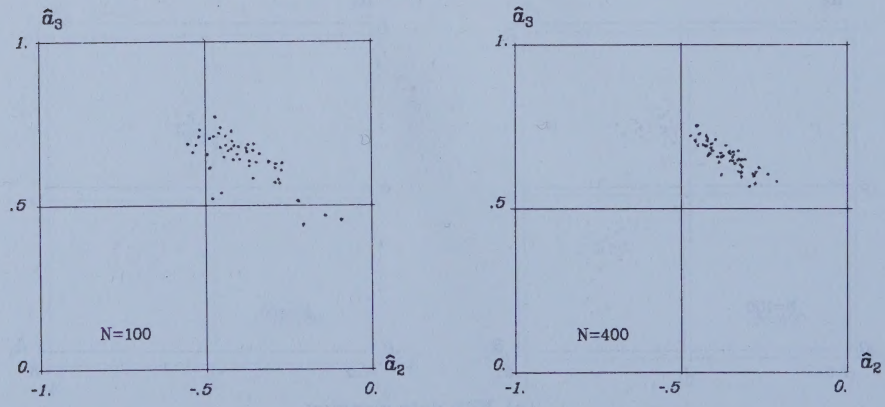
Figure 6. Estimates of LPC $\hat{a}_3$ v.s. $\hat{a}_1$ by the Burg algorithm in three cases: without outliers, with outliers, and after data cleaning



(a) Without outliers



(b) With outliers



(c) With data cleaning

Figure 7. Estimates of LPC $\hat{a}_3$ v.s. $\hat{a}_2$ by the Burg algorithm in three cases: without outliers, with outliers, and after data cleaning

RANDOMIZED ROBUST CONFIDENCE PROCEDURES

KENDALL MARTIN & MAX MINTZ
Department of Systems Engineering
University of Pennsylvania
Philadelphia, PA

INTRODUCTION & SUMMARY

This paper addresses the existence, structure, and behavior of <u>randomized</u> robust fixed size confidence procedures for a restricted parameter space. In particular, we consider the following location parameter estimation problem: Let Z denote a single observation of a scalar random variable, where $Z \in N(\theta,\sigma^2)$. Assume that θ is an unknown element of the given interval $\Omega = [-d, d]$, and that σ is an unknown element of $[\sigma_1, \sigma_2]$. Let A denote the action space of the statistician. Here, $A = [-d, d]$. Let $L(a,\theta)$ denote the zero-one loss function defined on $A \times \Omega$:

$$L(a,\theta) = 0, \quad |a-\theta| \leq e; \qquad (1)$$

$$L(a,\theta) = 1, \quad |a-\theta| > e;$$

where $e > 0$ is given. We wish to determine a robust minimax estimate $\delta^*(Z)$ for θ, based on these previous definitions and assumptions. Here, robustness refers to protection against the uncertainty in the nuisance parameter σ. We obtain randomized robust minimax admissible Bayes estimates for θ in this paper.

The connection between the robust minimax rule δ^* and a robust fixed size confidence procedure is obtained by noting that:

$$C^*(Z) = [\delta^*(Z)-e, \delta^*(Z)+e]$$

can be interpreted as a confidence procedure of size 2e which has the highest confidence coefficient $\inf_{\theta,\sigma} P_{\theta,\sigma}[\theta \in C^*(Z)]$.

The contribution of this present paper extends the results of [1,2] which focus on <u>nonrandomized</u> minimax and robust minimax rules for this decision paradigm. In particular, it is shown in [2] that if e is suitably large, then there exists a nonrandomized robust minimax rule. In this current paper, we show that the general solution to this problem -- arbitrary e and fixed d/e -- entails the consideration of randomized rules.

The present decision paradigm is quite relevant to engineering applications where one is faced with a "location" sensor whose signal to noise ratio can vary from measurement to measurement in some unknown fashion. These unknown fluctuations in sensor behavior could be due to: (i) a faulty or failing sensor, or (ii) a sensor which is subject to sporadic interference.

PRELIMINARIES

In order to identify the salient features of the underlying decision problem and obtain a brief exposition, we shall focus our primary attention in the sequel on the case where d = 3e.

In the remainder of this section we briefly review the solutions of two

decision problems which appear in [1,2]. These examples illustrate several concepts and results which provide the necessary background for the present paper.

Example 1: (Minimax Estimation [1]) Let $d = 3e$, and assume $\sigma_1 = \sigma_2 = \sigma$. Define a nonradomized odd monotone decision rule $\delta(Z)$ for $Z \geq 0$ by:

$$\delta(Z) = \begin{cases} 2e, & a+2e \leq Z; \\ Z-a, & a \leq Z < a+2e; \\ 0, & 0 \leq Z < a; \end{cases} \tag{2}$$

where $a \geq 0$ denotes a parameter to be specified in the sequel.

Let $R(\delta,\theta)$ denote the risk function of δ. For $\theta \geq 0$, $R(\delta,\theta)$ is:

$$R(\delta,\theta) = \begin{cases} F[(a-e)/\sigma], & e < \theta \leq 3e; \\ F[(-a-e)/\sigma], & \theta = e; \\ 2F[(-a-e)/\sigma], & 0 \leq \theta < e; \end{cases} \tag{3}$$

where F denotes the CDF of the standard normal distribution.

Due to the existing even symmetry of the loss function L and the normal sampling CDF $F[(z-\theta)/\sigma]$, and the odd symmetry of the decision rule $\delta(Z)$, it follows that the risk function R is an even function of θ.

We observe that R is piecewise constant over three nondegenerate subintervals of $[-3e, 3e]$. Let Regions I, II, and III denote respectively the subintervals $(-e, e)$, $(e, 3e]$, and $[-3e, -e)$. The minimax rule δ^* is obtained by selecting the parameter a to equalize the risk values over Regions I-III. Since R is even, this is equivalent to finding the parameter value a^* which solves: $F[(a^*-e)/\sigma] = 2F[(-a^*-e)/\sigma]$. It is shown in [1] that each δ defined by (2) is Bayes with respect to a prior density function of the form:

$$f(\theta) = k_1[U(\theta+e)-U(\theta-e)] + k_2[U(\theta+3e)-U(\theta-3e)], \tag{4}$$

where: $U(t) = 1$, if $t \geq 0$; $U(t) = 0$, if $t < 0$; and the parameters $\{k_i\}$ are nonnegative normalizing constants. The minimax property of δ^* is established by noting that: (i) δ^* is a Bayes rule with respect to a prior density f^*, and (ii) f^* assigns probability one to the Ω set on which $R(\delta^*,\theta)$ is maximized. Consider the following numerical example [1]: if $e = 0.1$, and $\sigma = 1$, then $a^* = 0.3992$, $k_1 = 0.1348$, $k_2 = 1.6217$, and the resulting minimax risk is 0.6176.

Definition 1: An estimate $\delta^*(Z)$ is said to be a robust minimax estimate for θ, if for all $\delta(Z)$:

$$\sup_{\theta,\sigma} R(\delta^*,\theta,\sigma) \leq \sup_{\theta,\sigma} R(\delta,\theta,\sigma).$$

Example 2: (Robust Minimax Estimation [2]) Let $d = 3e$, and assume that σ is an unknown element of $[\sigma_1, \sigma_2]$. Let $\delta(Z)$ denote a nonrandomized odd monotone decision rule of the form defined by (2). Let a^* denote the parameter value which solves: $F[(a^*-e)/\sigma_2] = 2F[(-a^*-e)/\sigma_2]$. It is shown in [2] that if $e \geq B$, where:

$$B = -(\sigma_2/2)F^{-1}[1/4], \tag{5}$$

then the estimate δ^* -- defined by (2) with $a = a^*$ -- is a robust minimax estimate. Let Ω' denote $\Omega \times [\sigma_1, \sigma_2]$. We observe that $R(\delta^*, \theta, \sigma)$ is given by (3), when $a = a^*$ and the domain Ω is replaced by Ω'. Let λ denote a probability measure on Ω. We can extend λ to Ω' by defining λ' to be the Ω-consistent measure which concentrates its mass on a cylinder set of the form $\Omega \times \{\sigma\}$. Let Λ denote the class of all probability measures λ on Ω with density functions defined by (4). Let Λ' denote the class of all probability measures λ' obtained by extending each $\lambda \in \Lambda$ to Ω'. It is shown in [2] that $\delta^*$is Bayes with respect to a prior distribution λ^* in the class delineated by Λ' with support defined by $\Omega \times \{\sigma_2\}$.

Remarks on Example 1 [1]:

Remark 1: We can properly restrict our attention to nonrandomized monotone decision rules. This follows by noting that the underlying decision problem is a monotone estimation problem in the sense of Karlin and Rubin [3,4,5]. In particular, we note that:

(a). $L(a',\theta)$ attains its minimum, as a function of a', at a point $a' = q(\theta)$, where q is a nondecreasing function of θ;

(b). $L(a',\theta)$, considered as a function of a', is nondecreasing as a' moves away from $q(\theta)$;

(c). A and Ω are closed intervals in E^1;

(d). the conditional distribution of Z given θ is nonatomic;

(e). the distribution of Z has a monotone likelihood ratio.

Conditions (a)-(e) above are sufficient to insure that the class of nonrandomized monotone decision rules is essentially complete.

Remark 2: The range space of the decision rules can be restricted to the interval $I = [-(d-e), (d-e)]$, i.e., $[-2e, 2e]$. This follows by noting that $L(d-e,\theta) = 0$, for $\theta \in [d-e, d]$, with a corresponding result for negative arguments.

Remark 3: The prior distribution $\lambda^* \in \Lambda$, which is delineated by the density function f^*, is a least favorable prior distribution.

Remarks on Example 2 [2]:

Remark 4: The minimax property of the Bayes rule δ^* is established by noting that: (i) if $e \geq B$, then $a^* \leq e$; and (ii) if $a^* \leq e$, then, for all θ, $R(\delta^*, \theta, \sigma)$ is maximized over σ at $\sigma = \sigma_2$; and as a consequence of (i)-(ii) the prior distribution $\lambda^* \in \Lambda'$ assigns probability one to the Ω'-set on which $R(\delta^*, \theta, \sigma)$ is maximized.

Remark 5: The prior distribution $\lambda^* \in \Lambda'$ is a least favorable prior distribution.

Remark 6: The natural parameter space Ω' in Example 2 is a subset of E^2. Thus, in this robust minimax estimation problem, the concepts of monotone likelihood ratio and monotone estimation do not apply. Hence, one might expect that the general solution (which includes the case where $e < B$) may require recourse to randomized decision rules.

RANDOMIZED ROBUST RULES

We begin this study of randomized procedures by focusing on a simplified version of the basic decision problem -- where $Z \in N(\theta,\sigma^2)$, $|\theta| \leq d$, $d = 3e$, $\sigma \in \{\sigma_1, \sigma_2\}$, and $\sigma_1 < \sigma_2$. Here, the uncertainty in σ has been reduced to a two-point uncertainty class. In order to extend the results of [2], we will only consider cases where $e < B$ (5).

Example 3: Let $e = 0.1$, $\sigma_1 = 0.5$, and $\sigma_2 = 1$. Here, $B = 0.337$, and therefore the inequality $e < B$ is satisfied.

Observation 1: If we restrict our attention to nonradomized monotone rules of the form (2), then a restricted minimax rule δ^* is obtained by solving:

$$\min_a \max\{ R(\delta,0,\sigma_1), R(\delta,0,\sigma_2), R(\delta,3e,\sigma_1), R(\delta,3e,\sigma_2) \}.$$

This computation is equivalent to solving:

$$F[(a^*-e)/\sigma_1] = 2F[(-a^*-e)/\sigma_2],$$

for a^*, and obtaining δ^* via (2). Here, $a^* = 0.3231$, and the maximum risk is 0.6722.

Observation 2: If we restrict our attention to randomized rules which are <u>independent</u> of Z, then a restricted minimax rule assigns equal probability to the actions $\{-2e, 0, 2e\}$. Here, the maximum risk is 2/3. Further, it is shown in [6] that the general minimax solution to the "blind" version of this decision problem can always be characterized by a discrete distribution with a finite number of atoms in $[-d, d]$.

Based on Observations 1 and 2, it is evident that a nonrandomized monotone rule, in the class delineated by (2), cannot provide a globally minimax solution in Example 3.

We next extend the class of rules defined by (2) to a class of randomized procedures. We begin by identifying certain desirable properties that this extended class of rules should exhibit.

Property P1: The randomized procedures should be realizable by a countable (but preferably finite) randomization over a class of nonrandomized elementary rules.

Property P2: The risk function R of each <u>non</u>randomized elementary rule should be piecewise constant in θ for fixed σ.

Property P3: The risk function R of each <u>randomized</u> rule should be an even function of θ for fixed σ. Thus, each randomized rule should exhibit odd symmetry.

Remarks on Properties P1-P3:

Remark 7: Initially, our justification for Properties P1 and P2 was based on their appealing simplicity, and our justification for Property P3 was based on the existing symmetry of the given decision problem.

A New Class of Elementary Rules:

Let D_0 denote the class of rules delineated by (2). Let D_1 denote the class of all nonrandomized monotone rules $\delta(Z)$:

$$\delta(Z) = \begin{cases} 2e, & b_1+2e \leq Z; \\ Z-b_1, & b_1 \leq Z < b_1+2e; \\ 0, & -b_2 \leq Z < b_1; \\ Z+b_2, & -b_2-2e \leq Z < -b_2; \\ -2e, & Z < -b_2-2e; \end{cases} \tag{6}$$

where b_1, b_2 denote parameters which satisfy the inequality: $-b_2 \leq b_1$.

Remarks on D_1:

Remark 8: D_0 is a proper subclass of D_1. Further, each $\delta' \in D_1$ can be represented by $\delta'(Z) = \delta(Z-s)$, where $\delta \in D_0$ and s denotes a shift parameter.

Remark 9: D_1 obeys Property P2, since if $\delta \in D_1$, then $R(\delta,\theta,\sigma)$ is:

$$R(\delta,\theta,\sigma) = \begin{cases} F[(b_1-e)/\sigma], & \theta \in (e, 3e]; \\ F[(-b_1-e)/\sigma], & \theta = e \\ F[(-b_1-e)/\sigma] + F[(-b_2-e)/\sigma], & \theta \in (-e, e); \\ F[(-b_2-e)/\sigma], & \theta = -e \\ F[(b_2-e)/\sigma], & \theta \in [-3e, -e). \end{cases} \tag{7}$$

Remark 10: If σ is known a priori, then each $\delta \in D_1$ is Bayes with respect to a prior density function of the form:

$$f(\theta) = \begin{cases} K_1', & \theta \in (e, 3e]; \\ K_2', & \theta \in [-e, e]; \\ K_3', & \theta \in [-3e, -e). \end{cases} \tag{8}$$

The Bayes property of the rules in D_1 is established by applying techniques which are similar to those in Appendix 1 of [1].

A Class of Prototype Randomized Rules D_2:

Let $\gamma(b_1,b_2,Z) \in D_2$ denote the randomized rule:

$$\gamma(b_1,b_2,Z) = \begin{cases} \delta(b_1,b_2,Z), & \text{with probability } 1/2; \\ -\delta(b_1,b_2,-Z), & \text{with probability } 1/2; \end{cases} \tag{9}$$

where: $\delta(b_1,b_2,Z) \in D_1$ (6).

Our initial reasoning for selecting the prototype randomized rule γ defined by (9) was based on the following observations and remarks:

Observation 3: If $\delta(Z)$ denotes any decision rule (measurable function) and $\delta_r(Z)$ denotes $-\delta(-Z)$, then based on the existing symmetry of the given decision problem:

$$R(\delta,\theta,\sigma) = R(\delta_r,-\theta,\sigma).$$

Observation 4: If $\delta(Z)$ denotes any decision rule, then:

$$\sup_{\theta} R(\gamma,\theta,\sigma) \leq \sup_{\theta} R(\delta,\theta,\sigma),$$

where:

$$\gamma(Z) = \begin{cases} \delta(Z), & \text{with probability } 1/2; \\ -\delta(-Z), & \text{with probability } 1/2. \end{cases}$$

Remark 11: Each prototype randomized rule γ defined by (9) obeys Properties P1 and P3.

Remark 12: Each prototype randomized rule γ defined by (9) has a risk function $R(\gamma,\theta,\sigma)$ which is symmetric in θ. For $\theta \geq 0$, R is:

$$R(\gamma,\theta,\sigma) = \begin{cases} [F[(b_1-e)/\sigma] + F[(b_2-e)/\sigma]]/2, & \theta \in (e, 3e]; \\ [F[(-b_1-e)/\sigma] + F[(-b_2-e)/\sigma]]/2, & \theta = e; \\ [F[(-b_1-e)/\sigma] + F[(-b_2-e)/\sigma]], & \theta \in [0, e). \end{cases} \tag{10}$$

D_3 -- A Larger Class of Randomized Rules:

Definition 2: Let D_3 denote the class of randomized rules obtained by randomizing over at most countably many randomized prototype rules in the class delineated by (9).

Remark 13: D_3 obeys Properties P1 and P3.

In the following section the class of randomized rules D_3 will be shown to contain a robust minimax rule for Example 3.

A SADDLE-POINT RESULT

In order to establish the existence of a robust minimax rule $\delta^* \in D_3$ for Example 3, we will first consider the following statistical game:

Let $\delta \in D_2$ (9) and define:

$$\begin{aligned} G_1(\underline{b}) &= R(\delta,3e,\sigma_2); \\ G_2(\underline{b}) &= R(\delta,0,\sigma_2); \\ G_3(\underline{b}) &= R(\delta,3e,\sigma_1); \\ G_4(\underline{b}) &= R(\delta,0,\sigma_1); \end{aligned} \tag{11}$$

where $\underline{b}$ denotes the parameter pair (b_1,b_2).

Let the triple (K, B, T) denote an abstract game where:

$T \subset E^3$ denotes the set of all four-point probability distributions delineated by (t_1,t_2,t_3,t_4);

$B \subset E^2$ denotes the set of all coefficients $\{(b_1,b_2): -b_2 \leq b_1\}$

which parameterize the prototype randomized rules (9); and

K: $B \times T \to E^1$ denotes the kernel function:

$$K = -\sum_{i=1}^{4} t_i G_i . \qquad (12)$$

We next investigate the existence of a saddle-point solution for the game (K, B, T), where we seek to maximize over $\underline{b} \in B$ and minimize over $\underline{t} \in T$.

Theorem 1: The game (K, B, T) admits a saddle-point solution which is delineated by a pure minimax strategy $\underline{t}^* \in T$, and an ε-maximin strategy q^*, which is a probabilistic mixture of at most four pure strategies in B.

Proof: The proof follows by noting that (K, B, T) satisfies a well-known minimax theorem due to Blackwell and Girshick [7]. Specifically, we note that:

(1). T is a compact convex subset of E^3;

(2). $K(\underline{b},\underline{t})$ is, for each $\underline{b} \in B$, a continuous convex function on T;

(3). $K(\underline{b},\underline{t})$ is, for each $\underline{t} \in T$, a continuous function on $B \subset E^2$.

Corollary 1 [7]: If B is replaced by a compact subset $\tilde{B} \subset B$, then the ε-maximin strategy q^* in Theorem 1 becomes a maximin strategy which can be realized by a mixture of at most four pure strategies in $\tilde{B}$.

The game $(K, \tilde{B}, T)$ is related to the minimax estimation problem in Example 3 through the following observations:

Observation 5: In Example 3 the strategy space B can be replaced by a compact subset, e.g., where $|b_i| \leq 10$.

Observation 6: The maximin solution of $(K, \tilde{B}, T)$ is equivalent to the minimax solution to:

$$\min_{\delta \in D_3} \max_{\substack{\theta \in \Omega \\ \sigma \in \{\sigma_1, \sigma_2\}}} R(\delta, \theta, \sigma).$$

Thus, we have obtained the existence of a minimax rule δ^* in the class of randomized rules D_3, as well as the existence of jointly least favorable prior distribution for the pair (θ, σ). Here, δ^* is obtained by randomizing over at most four rules in the class of prototype rules D_3.

In order to complete this analysis it is necessary to establish that the decision rule δ^* obtained in Observation 6 is globally minimax. The global minimax behavior of δ^* is addressed in the sequel by using a combination of decision-theoretic and computational techniques.

COMPUTATIONAL BAYES ANALYSIS

In order to establish that δ^* is globally minimax, it is sufficient to show that δ^* is Bayes with respect to a prior distribution λ^* which assigns probability one to the Ω'-set on which $R(\delta^*, \theta, \sigma)$ attains its maximum value.

Step 1: We first consider the family Ψ of prior distributions on Ω' which have set-wise uniform probability mass distributions on the following four subsets:

$$S_1 = [-3e, -e) \cup (e, 3e] \times \{\sigma_2\};$$

$$S_2 = [-e, e] \times \{\sigma_2\};$$

$$S_3 = [-3e, -e) \cup (e, 3e] \times \{\sigma_1\};$$

$$S_4 = [-e, e] \times \{\sigma_1\}.$$

We next compute numerically the least favorable prior distribution ψ^* in Ψ.

Based on this numerical computation, we find ψ^* to be:

$$\Pr[S_1] = 0.552;$$

$$\Pr[S_2] = 0.368;$$

$$\Pr[S_3] = 0.080;$$

$$\Pr[S_4] = 0.$$

The associated maximin risk is found to be: 0.6254.

Step 2: We first perform a minimax computation (via a search algorithm) with respect to the rules in the class D_2. This provides us with the simplest approximation to δ^*. The risk function and corresponding parameters of this first approximation δ_1 are:

$$R(\delta_1,\theta,\sigma) = 0.6289, \qquad (\theta,\sigma) \in S_1;$$

$$R(\delta_1,\theta,\sigma) = 0.6289, \qquad (\theta,\sigma) \in S_2;$$

$$R(\delta_1,\theta,\sigma) = 0.6289, \qquad (\theta,\sigma) \in S_3;$$

$$R(\delta_1,\theta,\sigma) = 0.5906, \qquad (\theta,\sigma) \in S_4;$$

$$\underline{b}^{(1)} = (1.2736, -0.2107).$$

We next perform a minimax computation with respect to the rules in the class obtained by randomizing over at most two rules in D_2. This provides us with a more accurate approximation to δ^*. The risk function and corresponding parameters of this second approximation δ_2 are:

$$R(\delta_2,\theta,\sigma) = 0.6273, \qquad (\theta,\sigma) \in S_1;$$

$$R(\delta_2,\theta,\sigma) = 0.6273, \qquad (\theta,\sigma) \in S_2;$$

$$R(\delta_2,\theta,\sigma) = 0.6250, \qquad (\theta,\sigma) \in S_3;$$

$$R(\delta_2,\theta,\sigma) = 0.5527, \qquad (\theta,\sigma) \in S_4;$$

$$\underline{b}^{(1)} = (1.5111, -0.0596); \qquad \underline{b}^{(2)} = (0.0676, -0.0321);$$

$$\Pr[\underline{b}^{(1)}] = 0.6115; \qquad \Pr[\underline{b}^{(2)}] = 0.3885.$$

Since the true global minimax risk must be in the interval [0.6254, 0.6273], we can terminate this approximation process at this stage with a maximum risk error of 0.0019 (0.304%).

If one wishes to continue this approximation and verification process, it is useful to note that one need only consider rules obtained by randomizing over at most three rules in D_2. This reduction in maximum complexity is a consequence of the fact that ψ^* assigns zero probability mass to the set S_4. Hence, the natural dimension of T can be reduced by one. This reduction can also be considered a consequence of the fact that, for $\delta \in D_2$, the value of $R(\delta, \theta, \sigma)$ on S_4 is dominated by the values on S_1-S_3.

FURTHER RESEARCH

In this paper we have determined the existence, structure, and behavior of a class of robust randomized decision rules. Since the applications paradigm (Example 3) is somewhat limited, it would be useful to to examine the efficacy of the rules in D_3 in less restricted decision settings. Extensions of this research currently in progress embrace decision paradigms with more extensive uncertainty classes, with specific attention being paid to non-Gaussian sampling distributions. These results will appear in [8].

REFERENCES

[1] Zeytinoglu, M. and Mintz, M. (1984). Optimal Fixed Size Confidence Procedures for a Restricted Parameter Space. Ann. Statist. 12, No. 3.

[2] Zeytinoglu, M. and Mintz, M. (1984). Robust Fixed Size Confidence Procedures for a Restricted Parameter Space. Submitted for publication to Ann. Statist.

[3] Karlin, S. and Rubin, H. (1956). The Theory of Decision Procedures for Distributions with Monotone Likelihood Ratio. Ann. Math. Statist. 27, 272-299.

[4] Berger, J. O. (1980). Statistical Decision Theory. Springer-Verlag, New York.

[5] Brown, L. D. (1976). Notes on Statistical Decision Theory. (Unpublished lecture notes, Ithaca.)

[6] Egerland, W. (1979). A Game-Theoretic Solution to an Engagement Problem. US Army Ballistic Research Laboratory Technical Report ARBRL-TR-02132. AD: A 066106.

[7] Blackwell, D. H. and Girshick, M. A. (1954). Theory of Games and Statistical Decisions. Wiley, New York.

[8] Martin, K. and Mintz, M. (1984). Randomized Robust Confidence Procedures. To be submitted for publication.

ACKNOWLEDGMENT

This work was supported in part by the following grants and contracts: the United States Air Force under Contract F33615-83-C-3000, and Grant AFOSR-82-NM-299; the National Science Foundation under Grant MCS-82-07294.

STABLE LINEAR DETECTORS WITH OPTIMALITY CONSTRAINTS

P. A. KELLY and W. L. ROOT
Computer, Information and Control Engineering
and the Aerospace Engineering Department
The University of Michigan
Ann Arbor, Michigan 48109-2140

ABSTRACT

This paper is concerned with the stable linear detection of deterministic signals in nominally Gaussian noise. Stability is measured in terms of the bounds on the distribution of the test statistic as the actual measure on the observation space varies in a Prokhorov neighborhood of the nominal; the derived stability index is independent of the size of the neighborhood. An optimization problem is formulated and solved to give the most stable linear detector meeting a constraint on the nominal level of detectability.

I. INTRODUCTION

Let H denote either R^N or $L_2[0,T]$, with Borel sets $B(H)$, and suppose that an observation $z = w+bs \in H$ is made. A standard problem is to decide between the hypotheses $b=0$ (hypothesis 0) and $b > 0$ (hypothesis 1) when w is modelled as additive noise distributed according to a mean-zero Gaussian measure with positive definite covariance Λ, and s, a known signal, is an element of the range of $\Lambda^{1/2}$. Let $\{\lambda_j\}$ be the eigenvalues of Λ and let $\{\phi_j\}$ be a corresponding sequence of orthonormal eigenvectors. Assume that the noise is scaled and the eigenvalues ordered so that $1=\lambda_1=\cdots=\lambda_k>\lambda_{k+1}\geq\cdots$, and that the signal is normalized so that

$$|| \Lambda^{-1/2} s || = 1 . \qquad (1)$$

Let $s_j = < s, \phi_j >$, $w_j = < w, \phi_j >$, and $z_j = < z, \phi_j >$. Then a uniformly most powerful test between hypotheses 0 and 1 compares $\Sigma(z_j s_j/\lambda_j)$ (the maximum likelihood estimate of b) to a threshold. (Note: throughout this paper, if no upper limit is placed on a summation sign, the sum is taken over the number of dimensions in H.)

Our problem is the following: Suppose the actual probability measure on H for a given b is only approximated by the nominal. We would like to make the test as insensitive as possible to deviations from the conditions of the model while maintaining a desired nominal level of detectability.

The most common approaches to robust design do not help here. The asymptotic minimax robustness results of Huber [1], El-Salwy and Vandelinde [2] , and Kelly [3], for example, are derived assuming that the noise components are iid, while those of Portnoy [4] and Poor [5] are based on assumptions of weak dependence and stationarity; all assume that the signal components are uniformly large. In those cases, laws of large numbers and central limit theorems are available for achieving asymptotic results. In our problem, the fact that s is in the range of $\Lambda^{1/2}$ means that we cannot use an asymptotic approach. In addition, while in the nominal problem the components of w are mutually independent, it seems to be unduly restrictive to require that they remain so or that they be stationary under a change in the measure on H . This suggests that the development of a nonlinear theory such as that of Huber [6] would be difficult. Finally, minimax techniques have the drawbacks of requiring specification of a level of uncertainty and of not necessarily guaranteeing good nominal performance.

A different approach is taken here. We consider statistics of the form $Hz = \langle z,h \rangle$ for some $h \in H$, and tests of the form:

$$\text{If } Hz \geq \tau \text{ , choose hypothesis 1 ;}$$
$$\text{otherwise, choose hypothesis 0 .} \qquad (2)$$

An indicator of how well this test performs under nominal conditions (an "optimality" index) is defined. Next, a stability index is determined which indicates how sensitive the statistic is to changes from the nominal. The problem of finding the most stable linear test satisfying a constraint on the optimality index is then solved. We note that, although the problem formulations are quite different, in some cases our solution is similar to the robust matched filter found by Poor [7] .

II. OPTIMALITY AND STABILITY INDEXES

In this section we derive expressions that characterize the optimality and the stability of a linear detector. First, suppose that the problem model is accurate. Assume further that the optimal detector for the model is not completely unstable; i.e., that weak convergence of measures on H to the nominal implies weak convergence of the distribution of the optimal statistic $\Sigma(z_j s_j / \lambda_j)$ to the nominal. As shown by Root [8], this implies that s is in the range of Λ. Then let the linear functional H_0 be given by $H_0 z = \langle z,\Lambda^{-1} s \rangle = \Sigma(z_j s_j/\lambda_j)$ and let the linear function H be given

by $Hz = \langle z,h \rangle = \Sigma z_j h_j$ for some $h \in H$. With E_0 and E_1 denoting expectation under hypotheses 0 and 1, respectively, we have:

$$E_0(H_0 z) = E_0(Hz) = 0$$
$$E_1(H_0 z) = b; \quad E_1(Hz) = b \Sigma s_j h_j \tag{3}$$
$$\sigma_0^2 \triangleq \mathrm{Var}(H_0 z) = 1; \quad \sigma^2 \triangleq \mathrm{Var}(Hz) = \Sigma \lambda_j h_j^2.$$

Whichever statistic is used in the test (2), pick τ so that the probability of false alarm is fixed at α_0; that is, $\tau = \sigma \Phi^{-1}(1-\alpha_0)$ (where Φ is the normal distribution). Then the optimal probability of detection is

$$\Phi(b-\tau) = \Phi(b-\Phi^{-1}(1-\alpha_0)) \tag{4}$$

while that achieved using the statistic Hz is

$$\Phi\left((b\Sigma s_j h_j - \tau)/\sigma\right) = \Phi(b(\Sigma\lambda_j h_j^2)^{-1/2}\Sigma s_j h_j - \Phi^{-1}(1-\alpha_0)). \tag{5}$$

This indicates that $(\Sigma\lambda_j h_j^2)^{-1/2} \Sigma s_j h_j$ (the square root of the nominal unit-amplitude signal-to-noise ratio) may be used as an optimality index. Since the normalization (1) implies that the optimal unit-amplitude SNR is 1, the optimality constraint on h is that, for some choice of $\delta > 0$,

$$I_0(h) \triangleq (\Sigma\lambda_j h_j^2)^{-1/2} \Sigma s_j h_j \geq (1-\delta) \quad . \tag{6}$$

Now consider the effect on detector performance of a change from nominal conditions. Let μ denote the nominal measure on H for a given b, and assume that the actual probability measure υ is such that the Prokhorov distance between μ and υ (denoted $\Pi(\mu,\upsilon)$) is $\leq \varepsilon$. $\Pi(\mu,\upsilon)$ is defined to be

$$\inf \{\eta : \upsilon(A) < \mu(A^\eta) + \eta \text{ for all } A \in B(H)\}$$

where $A^\eta \triangleq \{y: \inf_{x \in A} \| y-x \| \leq \eta\}$.

This characterization of uncertainty has been used in previous studies of robustness and stability (for example, Root [8] and Hampel [9]). A Prokhorov neighborhood of size ε includes all measures which fit the following common models for uncertainty: (i) ε-contaminations; (ii) total variations of size ε; and (iii) convolutions of the form $\upsilon(A) = \int_H \mu(A-x)d\omega(x)$, where ω is any probability measure placing a mass of at least $1-\varepsilon$ on an ε-neighborhood of the origin.

Let F_μ and F_υ be the distribution functions of Hz under the measures μ and υ, respectively. Then,

$$F_\upsilon(t) = \upsilon(\{ z: Hz \leq t \}) \leq \mu(\{ z: Hz \leq t \}^\varepsilon) + \varepsilon = \mu(\{ z: Hz \leq t + \varepsilon \| h \| \}) + \varepsilon = F_\mu(t + \varepsilon \| h \|) + \varepsilon \quad . \tag{7}$$

Similarly, $F_\upsilon(t) \geq F_\mu(t-\varepsilon \| h \|) - \varepsilon$. Now, $F_\mu(t)$ is equal to $\Phi((t-m)/\sigma)$, where m is the mean of Hz under μ. F_υ is a distribution, so we have

$$F_L(t) \leq F_\upsilon(t) \leq F_u(t)$$

where
$$F_L(t) \triangleq \max \{0, \Phi (\frac{t-m}{\sigma} - \varepsilon \frac{\| h \|}{\sigma}) - \varepsilon \} \tag{8}$$

and
$$F_u(t) \triangleq \min \{1, \Phi (\frac{t+m}{\sigma} + \varepsilon \frac{\| h \|}{\sigma}) + \varepsilon \} \tag{9}$$

(In fact, these bounds are tight in the sense that for any t_o there exist probability measures υ_L and υ_u on H such that $\Pi(\mu,\upsilon_L) \leq \varepsilon$ and $\Pi(\mu,\upsilon_u) \leq \varepsilon$ and such that $F_{\upsilon_L}(t) = F_L(t)$ for $t < t_o$ and $F_{\upsilon_u}(t) = F_u(t)$ for $t \geq t_o$. [10]). For any t, the greatest possible difference between Prob{Hz $\leq$ t|υ} and Prob{Hz $\leq$ t|μ} for υ with $\Pi(\mu,\upsilon) \leq \varepsilon$ is thus equal to $\max\{F_u(t) - F_\mu(t), F_\mu(t) - F_L(t)\}$. This motivates our characterizing the stability of the detector in terms of the quantity

$$K = \sup_t \max \{F_u(t) - F_\mu(t), F_\mu(t) - F_L(t)\} \quad . \tag{10}$$

By symmetry, K is in fact just $\sup_t\{F_u(t) - F_\mu(t)\}$, and it is clearly independent of m (and hence of the value of b). So,

$$K = \sup_t [\min\{1, \Phi (\frac{t}{\sigma} + \varepsilon \frac{\| h \|}{\sigma}) + \varepsilon \} - \Phi (\frac{t}{\sigma})] \quad . \tag{11}$$

It may be seen fairly easily that, for a fixed ε, K is a monotone increasing function of $\sigma^{-2} \| h \|^2$; so we choose as the stability index

$$I_s(h) \triangleq \sigma^{-2} \| h \|^2 = (\Sigma \lambda_j h_j^2)^{-1} \Sigma h_j^2 \tag{12}$$

Note that $I_s(h)$ does not depend on the size of the assumed uncertainty neighborhood.

III. MOST STABLE LINEAR DETECTOR

Our goal is to find the most stable detector meeting an optimality constraint. So, from (6) and (12), we have the problem:

Given $\delta > 0$, find $h \in H$ which minimizes $(\Sigma\lambda_j h_j^2)^{-1} \Sigma h_j^2$ subject to $(\Sigma\lambda_j h_j^2)^{-1/2} \Sigma s_j h_j \geq 1-\delta$. (I)

Neither $I_0(h)$ nor $I_s(h)$ are affected by multiplying h by a positive constant, so an equivalent problem to (I) is:

Given $\delta > 0$ find $h \in H$ which minimizes Σh_j^2 subject to $\Sigma s_j h_j \geq 1-\delta$ and $\Sigma\lambda_j h_j^2 = 1$. (II)

If h is a solution to problem (II) then for any constant $c > 0$, ch is a solution to problem (I). Now we make one more modification; let M be a positive integer less than or equal to the dimension of H, and consider the problem:

Given $\delta > 0$, find $h^M \in R^M$ which minimizes $\sum_M (h_j^M)^2$ subject to $\sum_M s_j h_j^M \geq 1-\delta$ and $\sum_M \lambda_j (h_j^M)^2 = 1$. ($II_M$)

Problem (II_M) can have a solution only if $(\sum_M s_j^2/\lambda_j)^{1/2} \geq 1-\delta$, so assume that M (or δ) is large enough that this is true. Define

$$r_M(x) = \sum_M \frac{\lambda_j s_j^2}{(1+x\lambda_j)^2} \Big/ \left(\sum_M \frac{s_j^2}{1+x\lambda_j} \right)^2 \tag{13}$$

and

$$q_M = r_M(0) = \sum_M \lambda_j s_j^2 \Big/ \left(\sum_M s_j^2 \right)^2 \tag{14}$$

Recall that $\lambda_j = 1$ for $j \leq K$, $\lambda_j < 1$ for $j > K$, and assume that $|s_j| > 0$ for each j (i.e., that each j for which $s_j = 0$ has been deleted). We have the result:

Theorem. The solution to problem (II_M) has the following form:

(a) If $\sum_L s_j^2 \geq (1-\delta)^2$, $L = \min(K,M)$: (15)

a minimizing h^M is given by

$$h_j^M = s_j \Big/ \left(\sum_L s_j^2 \right)^{1/2} \quad , \ j \leq L$$
$$= 0 \quad , \ j > L \text{ if } L < M. \tag{16}$$

In this case (the only possible one when $M \leq K$), $\sum_M (h_j^M)^2 = 1$. The solution is not unique unless the condition (15) is satisfied with equality.

(b) If $\sum_{j}^{K} s_j^2 < (1-\delta)^2$:

There is a unique global minimum provided by h^M as given by

$$h_j^M = \beta_M s_j/(1+\alpha_M\lambda_j) \qquad j = 1,\ldots,M \tag{17}$$

where

$$\beta_M = (1-\delta)\{\sum^{M}[s_j^2/(1+\alpha_M\lambda_j)]\}^{-1}$$

and where α_M is the unique number greater than -1 for which $r_M(\alpha_M) = (1-\delta)^{-2}$. The minimum, $\sum^{M}(h_i^M)^2$, is greater than 1. Further,

(b_1) if $q_M < (1-\delta)^{-2}$, then $-1 < \alpha_M < 0$;

(b_2) if $q_M = (1-\delta)^{-2}$, then $\alpha_M = 0$;

(b_3) if $q_M > (1-\delta)^{-2}$, then $\alpha_M > 0$.

Proof. Problem (II_M) has a solution (it is a problem of minimizing a continuous function over a compact, non-empty subset of R^M). The Kuhn-Tucker necessary conditions for h^M to be a solution are that there exist numbers α_M and β_M such that

$$h_j^M + \alpha_M\lambda_j h_j^M - \beta_M s_j = 0 \quad \text{for } j = 1,\ldots,M$$

with $\beta_M \geq 0$ and $\beta_M(1-\delta-\sum^{M} s_j h_j^M) = 0$. (See Theorem 4.3.6 in [11].)

(a) Suppose $\beta_M = 0$; then $h_j^M(1+\alpha_M\lambda_j) = 0$, $j = 1,\ldots,M$. That is, h_j^M must be zero for all $j \notin J(\rho)$, where $J(\rho)$ is the set of indices for which $\lambda_j = \rho$ for some ρ . Suppose we have a solution like this for $\rho < 1$. Then $\sum^{M} s_j h_j^M = 1-\delta_o \geq 1-\delta$, and $\sum^{M} \lambda_j (h_j^M)^2 = \sum_{J(\rho)} \rho(h_j^M)^2 = 1$, implying that $\sum_{J(\rho)} (h_j^M)^2 = \sum^{M}(h_j^M)^2 = \rho^{-1}$. Let $j_o \in J(\rho)$, with $(h_{j_o}^M)^2 = u^2 > 0$ and $u\, s_{j_o} > 0$, and let $0 < v < |u|$. Define $g \in R^M$ by $g_1 = \pm \rho^{-\frac{1}{2}}v$, $g_{j_o} = \pm (u^2-v^2)^{\frac{1}{2}}$, and $g_j = h_j^M$ for all $j \neq 1$ or j_o (with the signs of g_1 and g_{j_o} chosen to match those of s_1 and s_{j_o}, respectively). Then

$$\sum^{M}\lambda_j g_j^2 = \rho v^2 + 1 - \rho v^2 = 1 ,$$

$$\sum^{M} g_j^2 = \rho v^2 + \rho^{-1}(1-\rho v^2) = \rho^{-1} - v^2(1-\rho)$$

and

$$\sum^{M} s_j g_j = \rho^{-\frac{1}{2}}v|s_1| + 1-\delta_o - u\, s_{jo} + (u^2 - v^2)^{\frac{1}{2}}\,|s_{jo}|$$

For g to be a feasible solution to problem (II_M), we require that $\sum_{j}^{M} g_j s_j - (1-\delta) \geq 0$; that is,

$$f(v) \overset{\Delta}{=} \rho^{\frac{1}{2}} v|s_1| + (u^2-v^2)^{\frac{1}{2}} |s_{j_o}| + (\delta-\delta_o) - u\, s_{j_o} \geq 0 .$$

Note that $f(0) = \delta-\delta_o \geq 0$. Now,

$$\left.\frac{df(v)}{dv}\right|_{v=0} = \rho^{\frac{1}{2}} |s_1| - v(u^2-v^2)^{-\frac{1}{2}} |s_{j_o}| \Big|_{v=0}$$

$$= \rho^{\frac{1}{2}} |s_1| > 0 .$$

Thus, for some $v_o > 0$, $f(v_o) > 0$ and the g defined with $v=v_o$ is a feasible solution. But then

$$\sum_{j}^{M} g_j^2 = \rho^{-1} - v_o^2(1-\rho) < \rho^{-1} = \sum_{j}^{M} (h_j^M)^2$$

implying that h^M is not a minimizing solution.

From the above, a solution with $\beta_M = 0$ is possible only if $\rho = 1$; i.e., if $h_j^M = 0$ for $j > L$. Again, it is required that $\sum_{j}^{M} s_j h_j^M = \sum_{j}^{L} s_j h_j^M \geq 1-\delta$. By the Schwarz inequality,

$$(\sum_{j}^{L} s_j h_j^M)^2 \leq (\sum_{j}^{L} s_j^2/\lambda_j)(\sum_{j}^{L} \lambda_j (h_j^M)^2) = \sum_{j}^{L} s_j^2/\lambda_j = \sum_{j}^{L} s_j^2$$

so a solution is possible only if $\sum_{j}^{L} s_j^2 \geq (1-\delta)^2$. If this condition is met, let h^M be as given in (16). Then $\sum_{j}^{M} s_j h_j^M = (\sum_{j}^{L} s_j^2)^{\frac{1}{2}} \geq 1-\delta$ and $\sum_{j}^{M} \lambda_j (h_j^M)^2 = \sum_{j}^{M} (h_j^M)^2 = 1$. Since $\sum_{j}^{M} h_j^2 \geq 1$ for any h with $\sum_{j}^{M} \lambda_j h_j^2 = 1$, h^M is a global solution. That it is not unique unless $\sum_{j}^{L} s_j^2 = (1-\delta)^2$ follows from considering slight perturbations of h^M.

(b) Now suppose that $M > K$ and that $\sum_{j}^{K} s_j^2 < (1-\delta)^2$; by (a), we can only have a solution with $\beta_M > 0$ (so that $\sum_{j}^{M} s_j h_j^M = 1-\delta$). Then for each $j = 1,\ldots,M$, $h_j^M = \beta_M s_j/(1+\alpha_M\lambda_j)$. The constraints on h^M imply that $\beta_M \sum_{j}^{M}[s_j^2/(1+\alpha_M\lambda_j)] = 1-\delta$ and $\beta_M^2 [\sum_{j}^{M} \lambda_j s_j^2/(1+\alpha_M\lambda_j)^2] = 1$. That is,

$$\beta_M = (1-\delta)\{\sum_{j}^{M}[s_j^2/(1+\alpha_M\lambda_j)]\}^{-1} \tag{18}$$

and

$$r_M(\alpha_M) = (1-\delta)^{-2}. \tag{19}$$

Further, the assumption that there is no solution with $\sum_{j}^{M} s_j h_j^M > 1-\delta$ implies that any solution h^M must be such that h_j^M and s_j agree in sign for each j, so α_M cannot be less than -1. It is easy to check that r_M is a continuous function of x for $x > -1$, and that

$$\lim_{x\to\infty} r_M(x) = \Big(\sum^{M} s_j^2/\lambda_j\Big)^{-1} \qquad (< (1-\delta)^{-2} \text{ by assumption})$$

and

$$\lim_{x\downarrow -1} r_M(x) = \Big(\sum^{K} s_j^2\Big)^{-1} \qquad (> (1-\delta)^{-2} \text{ by assumption}).$$

In fact, the second limit is uniform in M. These limits and the continuity of r_M imply that there exists some $\alpha_M \in (-1,\infty)$ meeting the requirement (19). If $q_M = r_M(0) < (1-\delta)^{-2}$, then some $\alpha_M \in (-1,0)$ satisfies (19); if $q_M = (1-\delta)^{-2}$, then $\alpha_M = 0$ satisfies (19); and if $q_M > (1-\delta)^{-2}$, then some $\alpha_M > 0$ satisfies (19). Now, for $x > -1$, $(d\, r_M(x)/dx)$ has the same sign as

$$\Big\{\sum^{M} [\lambda_j s_j^2/(1+x\lambda_j)^2]\Big\}^2 - \Big\{\sum^{M} [s_j^2/(1+x\lambda_j)]\Big\}\Big\{\sum^{M} [\lambda_j^2 s_j^2/(1+x\lambda_j)^3]\Big\}$$

which an application of the Schwarz inequality shows to be strictly less than zero. Thus, r_M is a strictly monotone decreasing function on $(-1,\infty)$, so one and only one value of $\alpha_M > -1$ satisfies (19), implying the uniqueness of the solution (17) and completing the proof. ■

Of course, if $H = R^N$ for some finite N, the solution to problem (II_M) for $M = N$ is the solution to problem (II). If H is infinite-dimensional, consider the solutions to problems (II_M) as M increases. If $\sum^{K} s_j^2 \geq (1-\delta)^2$, then for all $M \geq K$, h^M has the form (16), and the minimizing $h \in H$ is obviously given by

$$h_j = s_j/\Big(\sum^{K} s_j^2\Big)^{1/2}, \quad j \leq K\ ;\ h_j = 0\ ,\ j > K\ .$$

Suppose that $\sum^{K} s_j^2 < (1-\delta)^2$; then for each M problem (II_M) has a solution h^M which depends on the constants α_M and β_M. Now, for each $x > -1$, it can be shown that $\{r_M(x)\}$ is a monotonically decreasing sequence for large M. This fact and the uniform in M convergence of $r_M(x)$ to $\big(\sum^{K} s_j^2\big)^{-1}$ as x decreases to -1 are used to prove the following:

Lemma. There exist finite numbers $\alpha > -1$ and $\beta > 0$ such that $\alpha_M \to \alpha$ and $\beta_M \to \beta$. Further, the α_M's decrease monotonically for large M.

Proof. See [10]. ■

With this lemma it is not difficult to show that a solution to problem (II) in the infinite-dimensional case is given by $h_j = \beta s_j/(1+\alpha\lambda_j)$, $j = 1,2,\ldots$. A proof is sketched in [12]; see [10] for details. We note, however, that in a sense the important results are contained in the above theorem and lemma: the finite-dimensional problems have computable solutions, and those solutions converge.

IV. MOST STABLE LINEAR ESTIMATOR

As a final point we mention the problem of stable linear estimation of

the amplitude parameter b. Let $\hat{b} = Hz = \Sigma z_j h_j$ for some $h = \{h_1, h_2, \ldots\} \in H$; this generalizes the nominal maximum likelihood estimate $\Sigma(z_j s_j/\lambda_j)$. It seems reasonable to require that $\hat{b}$ be nominally unbiased; from Eqs. (3), this implies that $\Sigma s_j h_j = 1$. The optimality of $\hat{b}$ is then a function of its variance which, from Eqs. (3), is equal to $\Sigma \lambda_j h_j^2$. This should be close to 1, and we write our optimality constraint as $\Sigma \lambda_j h_j^2 \leq (1-\delta)^{-2}$. If stability is defined in terms of the distance between the bounds on the distribution of b as the measure on H varies in Prokhorov neighborhood of the nominal, the stability index is again given by (12). The most stable nominally unbiased linear estimator of b meeting the optimality constraint is thus the solution to:

Given $\delta > 0$, find $h \in H$ which minimizes $(\Sigma \lambda_j h_j^2)^{-1} \Sigma h_j^2$ subject to $(\Sigma \lambda_j h_j^2)^{-\frac{1}{2}} \Sigma s_j h_j \geq 1 - \delta$ and $\Sigma s_j h_j = 1$. (III)

A comparison with problem (I) shows that, if h_o is a solution to problem (II), then a solution to problem (III) is $h = c_o h_o$, where $c_o = (\Sigma s_j h_{oj})^{-1}$.

ACKNOWLEDGEMENT

This research was supported in part by the National Science Foundation under Grant ECS-8005960.

REFERENCES

1. P.J. Huber, "Robust Estimation of a Location Parameter," Ann. Math. Stat., vol. 35, pp. 73-101, 1964.

2. A.H. El-Salwy and V.D. Vandelinde, "Robust Detection of Known Signals," IEEE Trans. Info. Theory, vol. IT-23, pp. 722-727, Nov. 1977.

3. P.A. Kelly, "Robust Estimation and Detection of Signals with Arbitrary Parameters," Proc. 21st Ann. Allerton Conference on Communication, Control, and Computing, pp. 602-609, 1983.

4. S.L. Portnoy, "Robust Estimation in Dependent Situations," Ann. Statist. vol. 5, pp. 22-43, 1977.

5. H.V. Poor, "Signal Detection in the Presence of Weakly Dependent Noise-Part II: Robust Detection," IEEE Trans. Info. Theory, vol. IT-28, pp. 744-752, Sept. 1982.

6. P.J. Huber, "A Robust Version of the Probability Ratio Test," Ann. Math. Stat., vol. 36, pp. 1753-1758, 1965.

7. H.V. Poor, "Robust Matched Filters," IEEE Trans. Info. Theory, vol. IT-29, pp. 677-687, Sept. 1983.

8. W.L. Root, "Stability in Signal Detection Problems," Proc. Symp. Appl. Math., vol. XVI, pp. 247-263, 1964.

9. F.R. Hampel, "A General Qualitative Definition of Robustness," Ann. Math. Stat., vol. 42, pp. 1887-1896, 1971.

10. P.A. Kelly, Ph.D. Dissertation, The University of Michigan, Ann Arbor, Michigan.

11. M.A. Bazaraa and C.M. Shetty, Nonlinear Programming: Theory and Algorithms. New York: Wiley, 1979.

12. P.A. Kelly and W.L. Root, "Stability in Detection of Signals in Noise," to appear in Proc. 1984 IEEE Conference on Decision and Control.

ON THE OPTIMALITY OF DATA PROCESSORS FOR SIGNAL DETECTION OVER A CLASS OF CONTAMINATED NOISES

D. R. HALVERSON
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843

and

G. L. WISE
Departments of Electrical and Computer Engineering and Mathematics
University of Texas at Austin
Austin, Texas 78712

ABSTRACT

We consider the effect induced on the data processor of a signal detection system when the underlying noise distribution functions are varied about their nominal values. We first consider the detection of a time varying deterministic signal in additive noise, and then extend our results to a more general situation in which the signal possesses a random amplitude. Our results characterize a class of contaminants of an arbitrary nominal distribution over which the data processor can be designed using the nominal distribution, and it is seen, for example, that such a class can contain distribution functions which can differ greatly from the nominal distribution.

I. INTRODUCTION

Consider the discrete time detection of a signal in corrupting noise. It is well known that a typical corresponding detector consists of a data processor followed by a threshold comparator. In fact, it is also well known that under a variety of fidelity criteria the data processor is characterized in terms of the relevant likelihood ratio. We note that it is often convenient in practice to employ an appropriate function of the likelihood ratio for the data processor. However, in order to present specific results about the general situation we will focus attention on the likelihood ratio itself.

An important factor which often limits the employment of the likelihood ratio within the data processor is inexact knowledge of the underlying statistical distributions. One way of resolving this difficulty is to employ a robust detector, such as the kind obtained from Huber's results (e.g. see [1]). Numerous authors have investigated various aspects of robust detection [2-6]. While robustness approaches can achieve the desired goal of desensitizing the data processors to inexact statistical knowledge, the would-be user of such detectors quickly becomes aware of their inherent limitations (for example, the extension of Huber's results to account for realistic forms of dependency and effects of nonstationarity, the question of the loss in performance occasioned by the robustness, etc.). For this reason, it is beneficial to consider situations where a simpler approach can be employed. In this paper we consider two commonly encountered situations; the first consists of the detection of a time varying deterministic signal in additive (not necessarily Gaussian) noise, whereas the second generalizes this situation to include a time varying signal with random amplitude. Our results allow determining when the likelihood ratio is invariant to perturbations in the underlying distributions from their nominal values, thus admitting the employment of the

nominal distributions to design the data processor.

II. DEVELOPMENT

The situation under consideration can be modeled as a choice between

H_0: μ_0,

H_1: μ_1

where μ_0 and μ_1 are finite measures defined on the Borel sets of $\mathbb{R}^n$. Let λ be a σ-finite dominating measure for μ_0 and μ_1, e.g. we could take $\lambda = \frac{1}{2}(\mu_0 + \mu_1)$. Consider first the detection of a time varying deterministic signal; that is, for a fixed element θ of $\mathbb{R}^n$ we set

$$\mu_1(B) = \mu_0(B-\{\theta\})$$

for all Borel sets B of $\mathbb{R}^n$. Note that the time varying signal θ is represented as a point in $\mathbb{R}^n$. We define the real, nonnegative, Borel measurable functions $f_0 = \frac{d\mu_0}{d\lambda}$ and $f_1 = \frac{d\mu_1}{d\lambda}$, and note that $f_1(\cdot) = f_0(\cdot-\theta)$ a.e. with respect to λ. We also let $S = \{x\epsilon\mathbb{R}^n: f_0(x)f_1(x) > 0\}$, and note that the likelihood ratio $\Lambda(\cdot)$ given by

$$\Lambda(x) = \frac{f_1(x)}{f_0(x)}$$

exists and is positive for all $x\epsilon S$. Note that S contains those elements of $\mathbb{R}^n$ for which we obtain a nondegenerate test; the cases where $\Lambda(x) = 0$ or $\Lambda(x) = \infty$ are easily checked.

Let $\mathcal{G}$ denote the class of all real, nonnegative, Borel measurable functions defined on $\mathbb{R}^n$; we will call an element of $\mathcal{G}$ a density. We then note that a wide class $\mathcal{C}$ of variations in the nominal density $f_0(\cdot)$ is generated by considering all densities $f: \mathbb{R}^n \to \mathbb{R}$ such that $f(\cdot) = (1-\varepsilon)f_0(\cdot) + \varepsilon g(\cdot)$, where $g\epsilon\mathcal{G}$ and ε is a real number satisfying $0 \leq \varepsilon < 1$ (cf. the ε-contamination class of [1]). In this case the actual density under H_0 is not necessarily the nominal density $f_0(\cdot)$ but rather an arbitrary element of $\mathcal{C}$, with an associated $g\epsilon\mathcal{G}$ and $\varepsilon\epsilon[0,1)$. In the following we consider only those g belonging to $\mathcal{G}$. We thus are actually testing between

H_0: $(1-\varepsilon)\ \mu_0 + \varepsilon\Gamma_0$,

H_1: $(1-\varepsilon)\ \mu_1 + \varepsilon\Gamma_1$,

where for each Borel set B in $\mathbb{R}^n$,

$$\Gamma_0(B) = \int_B g d\lambda \quad \text{and} \quad \Gamma_1(B) = \Gamma_0(B-\{\theta\}).$$

Equivalently, we test between the densities

H_0: $(1-\varepsilon)\ f_0(\cdot) + \varepsilon g(\cdot)$

H_1: $(1-\varepsilon)\ f_0(\cdot-\theta) + \varepsilon g(\cdot-\theta)$.

We are interested in the relevant likelihood ratio $\Lambda_{g,\varepsilon}(\cdot)$ given by, for $x \in S$,

$$\Lambda_{g,\varepsilon}(x) \triangleq \frac{f_0(x-\theta) + \varepsilon[g(x-\theta)-f_0(x-\theta)]}{f_0(x) + \varepsilon[g(x)-f_0(x)]} .$$

Note that $\Lambda_{g,\varepsilon}(x)$ exists for all $x \in S$ via our restrictions on ε. Note also that, for $x \in S$,

$$\frac{\partial \Lambda_{g,\varepsilon}(x)}{\partial \varepsilon} = \frac{[f_0(x) + \varepsilon(g(x)-f_0(x))][g(x-\theta)-f_0(x-\theta)]}{[f_0(x) + \varepsilon(g(x)-f_0(x))]^2}$$

$$- \frac{[f_0(x-\theta) + \varepsilon(g(x-\theta)-f_0(x-\theta))][g(x)-f_0(x)]}{[f_0(x) + \varepsilon(g(x)-f_0(x))]^2} .$$

We therefore have, for $x \in S$,

$$\frac{\partial \Lambda_{g,\varepsilon}(x)}{\partial \varepsilon} = 0 \quad \text{if and only if} \quad \frac{g(x-\theta)}{f_0(x-\theta)} = \frac{g(x)}{f_0(x)} .$$

This is true if and only if there exists a Borel measurable function $p: \mathbb{R}^n \to \mathbb{R}$ such that for all $x \in S$ we have $g(x) = p(x)f_0(x)$ and $p(x) = p(x-\theta)$. Since the above equivalence is true for all $\varepsilon \in [0,1)$, we have established the following result:

Theorem 1: Let A be a nonempty open subset of $[0,1)$ under the induced topology, and let $g \in \mathcal{G}$. We then have $\Lambda_{g,\varepsilon}(x) = \Lambda(x)$ for all $x \in S$ and for all $\varepsilon \in A$ if and only if there exists a Borel measurable function $p: \mathbb{R}^n \to \mathbb{R}$ such that for all $x \in S$,

$$g(x) = p(x)f_0(x) \quad \text{and} \quad p(x) = p(x-\theta).$$

Note that the above result characterizes the subclass of contaminants $g(\cdot)$ for which the likelihood ratio $\Lambda_{g,\varepsilon}(x)$ is invariant for each $x \in S$ as ε ranges over nonempty subsets which are open in $[0,1)$. This is true regardless of the nonempty open set A and in particular we can simply let ε be any number in $[0,1)$. The data processor of the detector could then be designed based on the nominal density with no degradation in performance for such contaminants g.

More generally, let us consider the extended situation where H_1 is nominally governed by the (Borel measurable) density $\tilde{f}_1(\cdot)$ given by

$$\tilde{f}_1(x) = \int_{\mathbb{R}^n} f_0(x-t)d\tilde{\mu}(t),$$

where for a fixed subset C of the nonzero integers the finite measure $\tilde{\mu}$ is defined on the Borel sets of $\mathbb{R}^n$ and is purely atomic with atoms $\{\{c\theta\}: c \in C\}$. Let $\tilde{S} = \{x \in \mathbb{R}^n: f_0(x)\tilde{f}_1(x) > 0\}$. This case includes the previous one and extends it to the situation where the signal is allowed to possess a certain form of random amplitude. As before, we consider variations in the nominal density $f_0(\cdot)$ which lie in $\mathcal{C}$. In particular, let $g \in \mathcal{G}$ be such that there exists a Borel measurable function $p: \mathbb{R}^n \to \mathbb{R}$ with $g(x) = p(x)f_0(x)$ and

$p(x) = p(x-\theta)$ for $x\epsilon\tilde{S}$. We then note that the likelihood ratio which corresponds to the nominal density $f_0(\cdot)$ is given on $\tilde{S}$ by

$$\tilde{\Lambda}(x) = \frac{\tilde{f}_1(x)}{f_0(x)},$$

whereas the one which corresponds to the variation $\tilde{f}(\cdot) = (1-\varepsilon)f_0(\cdot) + \varepsilon g(\cdot)$ is given, for all $x\epsilon\tilde{S}$ and for all $\varepsilon\epsilon[0,1)$, by

$$\tilde{\Lambda}_{g,\varepsilon}(x) = \frac{\tilde{f}_1(x) + \varepsilon \int_{\mathbb{R}^n} [g(x-t)-f_0(x-t)]d\tilde{\mu}(t)}{f_0(x) + \varepsilon[g(x)-f_0(x)]}$$

$$= \sum_{c\epsilon C} \frac{f_0(x-c\theta) + \varepsilon[g(x-c\theta)-f_0(x-c\theta)]}{f_0(x) + \varepsilon[g(x)-f_0(x)]} \cdot k_c$$

for some countable collection of real constants k_c which depend only on c, θ, and $\tilde{\mu}$. Note that it then follows from Theorem 1 that for each $x\epsilon\tilde{S}$, $\tilde{\Lambda}_{g,\varepsilon}(x) = \tilde{\Lambda}(x)$ for all $\varepsilon\epsilon[0,1)$. We have therefore established the following result:

Theorem 2: Suppose $g\epsilon\mathcal{G}$ and that there exists a Borel measurable function $p: \mathbb{R}^n \to \mathbb{R}$ with $g(x) = p(x)f_0(x)$ and $p(x) = p(x-\theta)$ for all $x\epsilon\tilde{S}$. It then follows that

$\tilde{\Lambda}_{g,\varepsilon}(x) = \tilde{\Lambda}(x)$ for all $x\epsilon\tilde{S}$ and for all $\varepsilon\epsilon[0,1)$.

The above result is thus an extension of Theorem 1 to a more general situation; however, we retain only the sufficiency condition. This, of course, is the condition we often might want to employ in practice.

Note that a key element in both Theorem 1 and Theorem 2 is that a contaminant which is a periodic multiple (with period equal to the signal) of the nominal density leads to an unperturbed likelihood ratio. The results of Theorem 1 and Theorem 2 provide us with an easily used tool to construct remarkedly unintuitive examples of noise contaminants, all of which employ the same optimal data processor for a wide variety of fidelity criteria. For example, let $n = 1$ and $\theta = 0.1$. If

$$f_0(x) = \frac{1}{2} I_{[-1,1]}(x) = \begin{cases} \frac{1}{2} & \text{if } x\epsilon[-1,1] \\ 0 & \text{otherwise} \end{cases},$$

i.e. the associated random variable is uniform on $[-1,1]$, and if the perturbation of f_0 is given by

$$f(x) = \begin{cases} 4.5\times10^{7} + 5\times10^{-2} & \text{if } x\epsilon[\frac{i}{10} - \frac{10^{-9}}{2}, \frac{i}{10} + \frac{10^{-9}}{2}] \cap [-1,1] \\ & \text{for } i = -10,-9,\ldots,10 \\ 0 & \text{if } x\epsilon(-\infty,-1)\cup(1,\infty) \\ 5\times10^{-2} & \text{otherwise} \end{cases},$$

then the resultant likelihood ratios are the same over $S = [-0.9,1]$, even though the two densities are very dissimilar. Notice in this example that $f(x)$ is nowhere equal to $f_0(x)$ on the interval $[-1,1]$.

For another example, let $n = 1$, $\theta = 2$, and

$$f_0(x) = 0.05\, I_{[-10,10]}(x) = \begin{cases} 0.05 & \text{if } x\in[-10,10] \\ 0 & \text{otherwise} \end{cases} .$$

Let D be a Cantor set (see, for example, [7]) in [0,1] with Lebesgue measure $\lambda\in(0,1)$. If the perturbation of f_0 is given by

$$f(x) = \begin{cases} \frac{1}{20\lambda} & \text{if } x\in D+\{2i\} \text{ for } i=-5,-4,\ldots,4 \\ 0 & \text{if } x\in(-\infty,-10)\cup(10,\infty) , \\ \frac{1}{40-20\lambda} & \text{otherwise} \end{cases}$$

then the resultant likelihood ratios are the same over S = [-8,10], even though the Cantor set results in a perturbation of f which is obviously very dissimilar to f_0. Notice that the maximum of the perturbation f can be made to exceed any preassigned real number.

III. CONCLUSION

We have considered the effect induced on the data processor of a signal detection system when the underlying distribution functions are varied about their nominal values. Having noted the crucial role played by the likelihood ratio in the detection system, we have therefore focused attention on the variation in the likelihood ratio itself. When testing first for the presence of a time varying deterministic signal, and then in a more general situation, we showed that the likelihood ratio is invariant over a class of perturbations of the nominal noise distribution.

ACKNOWLEDGEMENT

This research was supported in part by the Air Force Office of Scientific Research under Grants AFOSR-81-0047 and AFOSR-82-0033.

REFERENCES

1. P. J. Huber, *Robust Statistics*, Chap. 10. New York: Wiley, 1981.
2. R. D. Martin and S. C. Schwartz, "Robust detection of a known signal in nearly Gaussian noise," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 50-56, Jan. 1971.
3. S. A. Kassam and J. B. Thomas, "Asymptotically robust detection of a known signal in contaminated non-Gaussian noise," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 22-26, Jan. 1976.
4. A. H. El-Sawy and V. D. VandeLinde, "Robust detection of known signals," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 722-727, Nov. 1977.
5. H. V. Poor, "Robust decision design using a distance criterion," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 575-587, September 1980.
6. H. V. Poor, "Signal detection in the presence of weakly dependent noise, part II: robust detection," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 744-752, September 1982.
7. B. R. Gelbaum and J. M. H. Olmsted, *Counterexamples in Analysis*, pp. 88-89. San Francisco: Holden-Day, 1964.

GEOMETRICAL APPROACH TO GENERALIZED LEAST SQUARES ESTIMATION WITH SYSTOLIC ARRAY PROCESSING

S. KALSON AND K. YAO
Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90024

ABSTRACT

Algorithms for least squares problems play an important role in signal processing, digital data channel equalization, adaptive antenna array, spectrum estimation, system identification, etc. A geometric approach will be given for deriving algorithms solving least squares filtering problems that can be mapped into a systolic array. Order and time update recursive equations are derived. A flow diagram of the systolic array and its processing cells are presented. Thus, due to pipelining and concurrent processing in the array, a high throughput is obtained.

I. INTRODUCTION

A geometric approach will be given for deriving algorithms solving least squares filtering problems that can be mapped into a systolic array. Briefly stated, a systolic array is a set of interconnected processing cells in which each cell (except for boundary cells) communicates with only neighboring cells (local communication). All cells are governed by a master clock and because of pipelining and concurrent processing, a high throughput is obtained. Systolic arrays can be realized in hardware by VLSI technology, and thus are attractive for real time signal processing. See [1] for a more complete discussion of systolic arrays.

The systolic array given here based on a geometrical approach allows the construction of time and order update equations solving the least squares problem; similar results are obtained in [2], in which some of the variables are updated in the time index instead of the order index. The advantage of a geometric approach is that it leads to simplified derivations of a large number of algorithms for both a growing memory, exponentially weighted cost criterion and a sliding window cost criterion. However, only the growing memory case will be given in this paper.

II. PROBLEM STATEMENT

We shall be concerned with the following least squares problem: Given the column vectors $\bar{y}$ and $\bar{x}^i$, i=0,1,2,...p, at time T, find the filtered output

$$y_e^p(T) = \sum_{i=0}^{p} g_i^p(T)[\bar{x}^i]_T \qquad (1a)$$

(where $[\ ]_T$ denotes the T^{th} component), such that the time varying filter weights $g_i^p(T)$, i=0,1,...p, satisfy:

$$||\bar{y} - \sum_{i=0}^{p} g_i^p(T)\bar{x}^{-i}||_T^2 = \min_{q_i} ||\bar{y} - \sum_{i=0}^{p} q_i \bar{x}^{-i}||_T^2 \tag{1b}$$

where the norm $||\ ||_T^2$ is for any vector $\bar{u}$

$$||\bar{u}||_T^2 = <\bar{u},\bar{u}>_T \tag{2}$$

with inner product

$$<\bar{u},\bar{v}>_T = \sum_{t=1}^{T} \mu^{T-t}[\bar{u}]_t^*[\bar{v}]_t \tag{3a}$$

$$0 < \mu < 1 \tag{3b}$$

The inner product (3) defines a so-called exponentially weighted growing memory cost criterion. Notice that $\sum_{i=0}^{p} g_i^p(T)\bar{x}^{-i}$ can be interpreted as the best (in the least squares sense) linear estimate of the first T components of $\bar{y}$.

III. UPDATE THEOREMS BASED ON A GEOMETRICAL APPROACH

We shall show how the geometric formulism found in [3] can be used to derive a systolic array type algorithm solving (1). In [3], the least squares lattice algorithm (LSLA) solves the more restrictive least squares problem where the $\bar{x}^{-i}$ are related by time shifts of a single $\bar{x}$ for all i=0,1,...p. That is,

$$\bar{x}^{-i} = z^{-i}\bar{x} \tag{4}$$

where the shift operator z is defined by

$$[z^{-i}\bar{x}]_t = \begin{cases} [\bar{x}]_{t-i} & t > i \\ 0 & t \le i \end{cases} \tag{5}$$

and (5) makes use of the pre-windowed assumption

$$[\bar{x}]_t = 0 \quad t \le 0 \tag{6}$$

The LSLA can also be generalized in which (4) may not hold for all i, such as a pole-zero equalizer filter (decision feedback equalizer) in which

$$\bar{x}^{-i} = z^{-i}\bar{x} \quad i=0,1,2,...m \tag{7a}$$

$$\bar{x}^{-i} = z^{-i+m+1}\hat{\bar{y}} \quad i=m+1,...p \tag{7b}$$

where $\bar{x}$ is the received message sequence and $\hat{\bar{y}}$ is the estimate of the transmitted message sequence, [4]. However, for the case in which (4) does not hold for all i, the LSLA involves matrix operations, whereas the systolic array given here involves only scalar operations.

Let the residual vector $\overline{\gamma}^p(T)$ be

$$\overline{\gamma}^p(T) = \overline{y} - \sum_{i=0}^{p} g_i^p(T)\overline{x}^i \tag{8}$$

Then we seek $\gamma^p(T) = [\overline{\gamma}^p(T)]_T$ since the filtered output $y_e^p(T)$ is

$$y_e^p(T) = [\overline{y}]_T - \gamma^p(T) \tag{9}$$

It is well known that (1) implies that $\overline{\gamma}^p(T)$ is orthogonal to $\overline{x}^i$ with respect to the inner product $<\ ,\ >_T$. That is,

$$<\overline{\gamma}^p(T),\ \overline{x}^i>_T = 0 \quad i=0,1,\ldots p \tag{10}$$

Let S^p be the linear vector space

$$S^p = \text{Lin span } \{\overline{x}^o,\ldots\overline{x}^p\} \tag{11a}$$

$$S^{-1} = \emptyset \text{ (null set)} \tag{11b}$$

Then from (8) and (10), $\overline{\gamma}^p(T)$ is the orthogonal projection of $\overline{y}$ onto the orthogonal complement of S^p with respect to $<\ ,\ >_T$. Denote this as,

$$\overline{\gamma}^p(T) = \overline{y} - E(S^p,T)\overline{y} \tag{12a}$$

$$= E(S_\perp^p,T)\overline{y} \tag{12b}$$

where $S_\perp^p$ is the orthogonal complement of S^p, and $E(S,T)$ is the projection operator of a vector onto the space S with respect to the inner product $<,>_T$. We now present two theorems.

Theorem 1. Order updates

For the inner product space S^p and for any vector $\overline{u}$, we have the following order update relation in the index p

$$E(S_\perp^p,T)\overline{u} = E(S_\perp^{p-1},T)\overline{u} - <\overline{\beta}^p(T),E(S_\perp^{p-1},T)\overline{u}>_T \frac{\overline{\beta}^p(T)}{\epsilon_\beta^p(T)} \tag{13}$$

where

$$\overline{\beta}^p(T) = E(S_\perp^{p-1},T)\overline{x}^p \tag{14a}$$

$$\epsilon_\beta^p(T) = ||\overline{\beta}^p(T)||_T^2 \tag{14b}$$

Theorem 2. Time updates

For any inner product space S with the growing memory inner product of (3a), we have for any vector $\overline{u}$ the time update relation

$$E(S_\perp,T)\overline{u} = E(S_\perp,T-1)\overline{u} - [E(S_\perp,T-1)\overline{u}]_T E(S,T)\overline{e}^T \tag{15}$$

where

$$[\overline{e}^T]_t = \begin{cases} 1 & t=T \\ 0 & t\neq T \end{cases} \tag{16}$$

and for any two vectors $\overline{u}$ and $\overline{v}$,

$$\langle E(S_\perp,T)\bar{u},E(S_\perp,T)\bar{v}\rangle_T = \mu\langle E(S_\perp,T-1)\bar{u},E(S_\perp,T-1)\bar{v}\rangle_{T-1} + [E(S_\perp,T-1)\bar{u}]_T^*[E(S_\perp,T)\bar{v}]_T \tag{17}$$

Theorem 1 follows easily from the fact that S^p can be decomposed as

$$S^p = \text{Lin span}\{S^{p-1},\bar{x}^p\} \tag{18a}$$
$$= S^{p-1}\oplus\bar{\beta}^p(T)$$

and thus the projection onto S^p is the sum of projections onto S^{p-1} and $\bar{\beta}^p(T)$,

$$E(S^p,T)\bar{u} = E(S^{p-1},T)\bar{u} + \langle\bar{\beta}^p(T),\bar{u}\rangle_T \frac{\bar{\beta}^p(T)}{\epsilon_\beta^p(T)} \tag{18b}$$

Since $E(\cdot,T)$ is idempotent and self adjoint,

$$\begin{aligned}\langle\bar{\beta}^p(T),\bar{u}\rangle_T &= \langle E(S_\perp^{p-1},T)\bar{x}^p,\bar{u}\rangle_T \\ &= \langle (E(S_\perp^{p-1},T)^2\bar{x}^p,\bar{u}\rangle_T \\ &= \langle E(S_\perp^{p-1},T)\bar{x}^p,E(S_\perp^{p-1},T)\bar{u}\rangle_T \\ &= \langle\bar{\beta}^p(T),E(S_\perp^{p-1},T)\bar{u}\rangle_T\end{aligned} \tag{18c}$$

and since $E(S_\perp^p,T) = I - E(S^p,T)$, Eq.(13) follows. To verify Theorem 2, we must have

$$\langle E(S_\perp,T)\bar{u},\bar{x}\rangle_T = 0 \quad \forall\bar{x}\in S \tag{19}$$

Substituting the right-hand side of (15) for $E(S_\perp,T)\bar{u}$ in (19) and using the fact that for the growing memory inner product

$$\langle\bar{u},\bar{v}\rangle_T = \mu\langle\bar{u},\bar{v}\rangle_{T-1} + [\bar{u}]_T^*[\bar{v}]_T \tag{20}$$

as well as

$$\langle E(S,T)\bar{e}^T,\bar{x}\rangle_T = [\bar{x}]_T \quad \forall\bar{x}\in S \tag{21}$$

it is possible to verify Eq.(19). Equation (17) follows from

$$\langle E(S_\perp,T)\bar{u},\ E(S_\perp,T)\bar{v}\rangle_T = \langle E(S_\perp,T)\bar{u},\bar{v}\rangle_T \tag{22}$$

and substituting (15), for $E(S_\perp,T)\bar{u}$ in (22). Theorem 2 is from [3], which can be referred to for a complete proof.

IV. RECURSIVE EQUATIONS

We now show how these theorems are used to derive an algorithm solving (1) and giving $\gamma^p(T)$. To this end, application of theorem 1 to (12) gives

$$\bar{\gamma}^p(T) = \bar{\gamma}^{p-1}(T) - k_\gamma^p(T)\frac{\bar{\beta}^p(T)}{\epsilon_\beta^p(T)} \tag{23}$$

where

$$k_\gamma^p(T) = \langle\bar{\beta}^p(T),\bar{\gamma}^{p-1}(T)\rangle_T \tag{24}$$

Taking the T^{th} component of both sides of (23) gives

$$\gamma^p(T) = \gamma^{p-1}(T) - k_\gamma^p(T) \frac{\beta^p(T)}{\epsilon_\beta^p(T)} \tag{25}$$

where

$$\beta^p(T) = [\overline{\beta}^p(T)]_T \tag{26}$$

The rational for deriving a recursion in p for $\gamma^p(T)$ is that $\gamma^{-1}(T)$ is already "known", i.e.,

$$\gamma^{-1}(T) = [\overline{y}]_T - [E(S^{-1},T)\overline{y}]_T \tag{27}$$

$$= [\overline{y}]_T$$

Thus, we need to find $k_\gamma^p(T)$, $\epsilon_\beta^p(T)$, and $\beta^p(T)$. A time update will be given for $k_\gamma^p(T)$ and order opdates are given for $\epsilon_\beta^p(T)$ and $\beta^p(T)$. The systolic array of [2] has time updates for $\epsilon_\beta^p(T)$. First, we derive order updates for $\overline{\beta}^p(T)$. This is obtained by defining the auxiliary vector

$$\overline{\beta}_j^p(T) = E(S_\perp^{j-1},T)\overline{x}^p \tag{28}$$

Notice that

$$\overline{\beta}_p^p(T) = \overline{\beta}^p(T) \tag{29a}$$

$$\overline{\beta}_o^p(T) = \overline{x}^p \tag{29b}$$

Thus, since $\overline{x}^p$ is known, $\overline{\beta}^p(T)$ (or $\beta^p(T)$) will be given by a recursive relation in the index j of $\overline{\beta}_j^p(T)$. This again is given by direct application of Theorem 1.

$$\beta_{j+1}^p(T) = \beta_j^p(T) - k_{jp}(T)\ \beta^j(T)/\epsilon_\beta^j(T) \tag{30}$$

where

$$k_{jp}(T) = \langle \overline{\beta}^j(T), \overline{\beta}_j^p(T) \rangle_T \tag{31a}$$

$$\beta_j^p(T) = [\overline{\beta}_j^p(T)]_T \tag{31b}$$

Order updates for $\epsilon_\beta^j(T)$ can now be obtained. Let

$$\epsilon_j^p(T) = ||\overline{\beta}_j^p(T)||_T^2 \tag{32}$$

Then

$$\epsilon_o^p(T) = ||\overline{x}^p||_T^2 \tag{33a}$$

$$\epsilon_p^p(T) = \epsilon_\beta^p(T) \tag{33b}$$

Since $\epsilon_o^p(T)$ is given by the time update equation

$$\epsilon_o^p(T) = \mu\epsilon_o^p(T-1) + |[\overline{x}^p]_T|^2 \tag{34}$$

$\epsilon_\beta^p(T)$ will be given by an order update in the index j of $\epsilon_j^p(T)$. From the self adjoint and idempotent properties of $E(\cdot,T)$, we have

$$\epsilon_j^p(T) = \langle \overline{\beta}_j^p(T), \overline{x}^p \rangle_T \tag{35}$$

and thus from (30), we have

$$\epsilon_{j+1}^{p}(T) = \epsilon_{j}^{p}(T) - |k_{jp}(T)|^2/\epsilon_{\beta}^{j}(T) \tag{36}$$

Time updates for $k_{\gamma}^{p}(T)$ and $k_{jp}(T)$ are derived by using Theorem 2. From Eq.(17),

$$k_{\gamma}^{p}(T) = \mu k_{\gamma}^{p}(T-1) + [\overline{\beta}^{p}(T-1)]_{T}^{*} \gamma^{p-1}(T) \tag{37}$$

However, we need to express $[\overline{\beta}^{p}(T-1)]_{T}$ in terms of $\beta^{p}(T)$. By using Theorem 1 and Eq.(15) of Theorem 2, it is possible to show that

$$[\overline{\beta}^{p}(T-1)]_{T} = \beta^{p}(T)/(1-\sigma^{p-1}(T)) \tag{38}$$

where $\sigma^{p}(T)$ is a positive real number less than one and is given by the order update equation

$$\sigma^{p}(T) = \sigma^{p-1}(T) + |\beta^{p}(T)|^2/\epsilon_{\beta}^{p}(T) \tag{39}$$

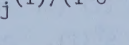
with $\sigma^{-1}(T) = 0$. A derivation can be found in [3]. Thus, the time update for $k_{\gamma}^{p}(T)$ is

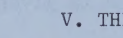
$$k_{\gamma}^{p}(T) = \mu k_{\gamma}^{p}(T-1) + \beta^{p}(T)^{*}\gamma^{p-1}(T)/(1-\sigma^{p-1}(T)) \tag{40}$$

Likewise, for $k_{jp}(T)$ we have

$$k_{jp}(T) = \mu k_{jp}(T-1) + \beta^{j}(T)^{*}\beta_{j}^{p}(T)/(1-\sigma^{j-1}(T)) \tag{41}$$

V. THE SYSTOLIC ARRAY

Now we consider the systolic array implementation of the recursive equations derived in the last section. These update equations appear somewhat messy, but the way in which they are combined can probably best be understood by considering the systolic array of Fig.1. A delay of i time units (clock cycles) is represented by 

, and a delay of one time unit by  . The three processing cells of Fig.1 are described in Figs.2,3, & 4. Notice that the time update equations (40) and (41) allow the processing cells to satisfy the local communication constraint.

The delay shifts are needed so that the correct data reach the processing cells at the correct time. For example, at T=1, $[\overline{x}^{o}]_{1}$ reaches the first circular cell of Figure 1. At T=2, $[\overline{x}^{-1}]_{1}$ and the results from the first circular cell, i.e., $\beta^{o}(1)$, $\epsilon_{\beta}^{o}(1)$, both reach the lower left hand square cell to be processed. Thus, "wavefronts" of intermediate results propagate through the array as shown in Fig.5 . The final results, i.e., $\gamma^{p}(T)$ (or $y_{e}^{p}(T)$), occur when the wavefronts exit through the boundary cells. Notice that successive wavefronts are next to each other. This is because the systolic array is fully pipelined, i.e., data enters the array for each clock cycle. The throughput rate of a fully pipelined array is the recipro-

cal of the clock cycle time. In the case of decision directed or decision feedback equalization, a wavefront of data must pass through most of the array, followed by a decision on the filtered output, before a new wavefront of data can be fed into the array. Thus, in such cases, the systolic array is not fully pipelined.

If y is a known training sequence, then the systolic array must be run as a fixed filter after the training of the equalizer. This is accomplished by freezing $k_\gamma^p(T)$, $k_{jp}(T)$, $\epsilon_j^p(T)$, and performing the update

$$y_e^p(T) = y_e^{p-1}(T) + k_\gamma^p(T)\beta^p(T)/\epsilon_\beta^p(T) \tag{42}$$

(with $y_e^{-1}(T)=0$), in the top row of processing cells in Fig.1. It must be pointed out that when the array is first "turned on", a division by zero will occur in some of the processing cells since $\sigma^p(T)=1$ and $\epsilon_\beta^p(T)=0$ for $p \geq T-1$. This happens because the least squares problem of Eq.(1) can be solved exactly with zero residual error. Thus, it is prudent to define a division by zero as equal to zero, so that the correct results, i.e., $\gamma^p(T)=0$, $\beta^p(T)=0$, are obtained.

VI. CONCLUSION

A geometrical approach to deriving algorithms for general least squares filtering is presented and a new systolic array is given. The geometrical approach leads to simplified derivations of algorithms that can be quite complex because of the interleaving of time and order updates. Although only the growing memory inner product is considered, it is also possible to derive algorithms for the sliding window inner product

$$\langle \bar{u}, \bar{v} \rangle_T = \sum_{t=T-l+1}^{T} [\bar{u}]_t^* [\bar{v}]_t \tag{43}$$

Also, algorithms for decision directed and decision feedback equalization can easily be derived, see [5].

III. ACKNOWLEDGEMENT

This work was partially supported by the electronics program of the Office of Naval Research.

REFERENCES

[1] Kung, H.T., and Leiserson, C.E., "Systolic Arrays (doe VLSI)," Sparse Matrix Proc. 1978, Duff, I.S., and Stewart, G.W., eds., SIAM 1979, pp.256-282.

[2] J.G. McWhirter, "Recursive Least-Squares Minimization Using a Systolic Array," Proc. SPIE, Vol.431, 1983, pp.

[3] M.J. Shensa, "A Least Squares Lattice Algorithm: A Geometrical Approach," IEEE Trans. Automat. Contr., Vol.AC-26, June 1981, pp.695-702.

[4] M.J. Shensa, "A Least-Squares Lattice Decision Feedback Equalizer," in Proc. IEEE ICC 1980, Seattle, WA, 1980, pp.57.6.1-57.6.5.

[5] S. Kalson, Ph.D. dissertation, University of California, Los Angeles, In preparation.

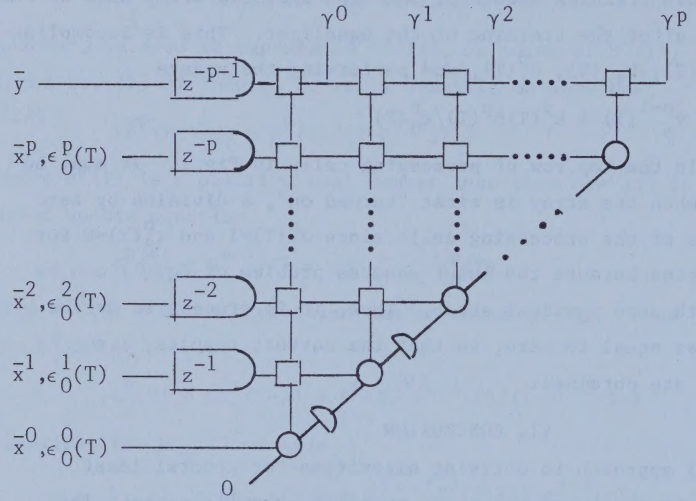
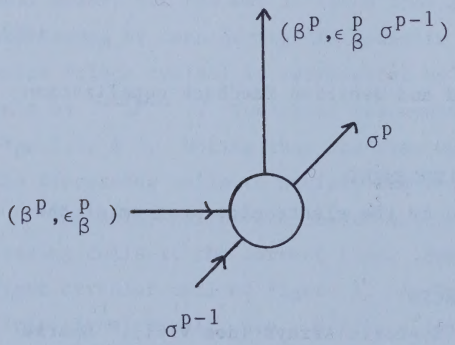


figure 1. Systolic array for least squares estimation

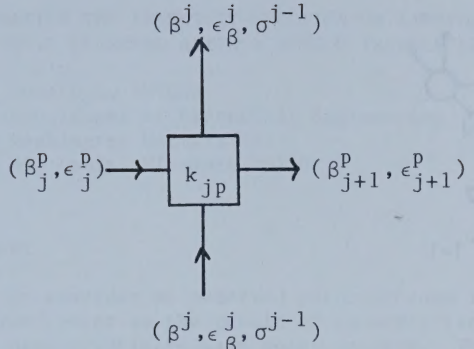


CELL DESCRIPTION:

For each clock cycle do,

$$\sigma^p = \sigma^{p-1} + \frac{|\beta^p|^2}{\epsilon_\beta^p}$$

figure 2. Cell descrioption for updating σ^p



CELL DESCRIPTION

k_{jp} is stored inside cell

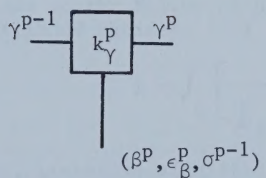
For each clock cycle do,

$$k_{jp} \leftarrow k_{jp} + \frac{\beta^{j*}\beta^p_j}{1-\sigma^{j-1}}$$

$$\beta^p_{j+1} = \beta^p_j - \frac{k_{jp}\beta^j}{\epsilon^j_\beta}$$

$$\epsilon^p_{j+1} = \epsilon^p_j - \frac{|k_{jp}|^2}{\epsilon^j_\beta}$$

figure 3. Cell description for β^p_j and ϵ^p_j



CELL DESCRIPTION:

k^p_γ is stored inside cell

For each clock cycle do:

$$k^p_\gamma \leftarrow \mu k^p_\gamma + \frac{\beta^{p*}\gamma^{p-1}}{1-\sigma^{p-1}}$$

$$\gamma^p = \gamma^{p-1} - \frac{k^p_\gamma \beta^p}{\epsilon^p_\beta}$$

If systolic array is to be run as a filter for the output y^p_e, then instead of updating γ^p, do:

$$y^p_e = y^{p-1}_e + \frac{k^p_\gamma \beta^p}{\epsilon^p_\beta}$$

figure 4. Cell description for $\gamma^p(y^p_e)$.

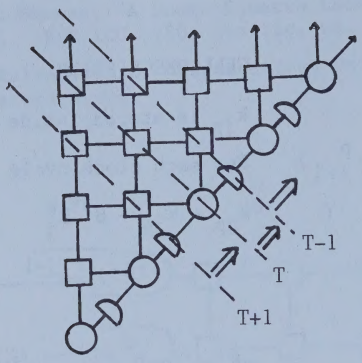


figure 5. Wavefront description

ESTIMATING THE INTENSITY FUNCTION OF A POISSON PROCESS WHEN EACH OF ITS POINTS IS OBSERVED AFTER A RANDOM TRANSLATION*

DONALD L. SNYDER
Department of Electrical Engineering
Washington University
St. Louis, Missouri 63130

SUMMARY

We consider an observed point-process in which the location of each observed point is the result of randomly translating a point of an underlying, unobservable Poisson point-process. The problem we address is that of estimating the intensity function of the underlying Poisson process.

Let $\{N(dx),\ x \in X\}$ be a Poisson point-process on a space X with intensity function $\{\lambda(x),\ x \in X\}$. Here, N(dx) is the number of points in [x,x+dx). Suppose a point of this process occurs at location x and that this point is moved randomly to location u in another space U according to the conditional probability density p(u/x). Each point of N is translated randomly and independently in this manner to produce an observed point-process M(du) on the space U. The following property of the observed point-process is readily established.

Property: The observed point-process M is a Poisson process with intensity $\mu(u)$ given by

$$\mu(u) = \int_X p(u|x)\lambda(x)m(dx),$$

where $m(\cdot)$ is the volume of [x,x+dx).

We wish to estimate $\{\lambda(x),\ x \in X\}$ given the observed point-process on U. This leads to the following nonlinear integral equation for the maximum-likelihood estimate $\hat{\lambda}$ of λ:

$$1 = \int_U \left[\frac{p(u|x)}{\int_X p(u|x)\hat{\lambda}(x)m(dx)} \right] M(du)$$

A convergent, iterative algorithm is identified for solving this integral equation numerically.

This estimation problem is motivated by an application in medical imaging of positron emitting radionuclides, where $X = R^2$ and $U = R^2 \times [0,\pi)$; see [1].

*This work was supported by the National Science Foundation under grant ECS-8215181 and by the National Institutes of Health under grants RR01379 and RR01380 from the division of Research Resources.

1. D.L. Snyder and D.G. Politte, "Image Reconstruction for List-Mode Data in an Emission-Tomography System Having Time-of-Flight Measurements," IEEE Transactions on Nuclear Science, Vol. NS-20, pp. 1843-1849, June 1983.

LARGE ERROR PERFORMANCE OF CROSS-CORRELATION TYPE ESTIMATORS OF TIME DELAY

ALFRED HERO
Dept. of EECS
The University of Michigan
Ann Arbor, Michigan 48109

STUART SCHWARTZ
Dept. of EECS
Princeton University
Princeton, New Jersey 08544

ABSTRACT

The large error performance of correlation methods in time delay estimation is considered using a level crossing model for peak ambiguity. The approach is sufficiently general to be applied to both broadband and narrowband signal spectra. Here the rate function of the large errors and a corresponding global variance approximation are derived and investigated for broadband spectra.

I. INTRODUCTION

Correlation methods of time delay estimation have been popular due to their simplicity and optimality for large observation times [1]. Of continuing interest is the performance of the estimate over a wide range of operating conditions. When the *observation time - signal bandwidth product* (BT) is large, a small error approximation to estimator variance can be applied for high *signal-to-noise ratio* (SNR) [6]. However, for small SNR or small BT, the approximation is known to significantly underestimate the true variance of the correlation estimate. For lowpass signal spectra and large BT, Ianniello [2] developed a large error probability approximation based on a discrete model for the erroneous estimates. The approximation [2] is implemented as an integral over a joint probability of a large number of discrete events. Even with an i.i.d. Gaussian assumption on these events the probabiltity calculation is tedious, involving numerical integration. In [3] Ianniello's probability approximation was used to approximate the global variance of the correlation estimate using an assumption of uniformly distributed error. The resulting variance expression was then shown to compare well with simulation results for lowpass signals and high BT.

In this paper we present an alternative method of approximating the large error performance of correlation type time delay estimates. By looking at the continuous time nature of the cross-correlation trajectory, a level crossing model for large errors is invoked and the rate function is calculated. Then, based on work reported elsewhere [4] on the distribution of level crossings, the global variance of the estimate is approximated by means of an inhomogeneous Poisson model for the errors. The resulting variance expression involves the small error variance and a correction term for large errors: the radius of gyration of the intensity function. The advantage of our approach is the elimination of tenuous uniformity assumptions on the values of the deviant estimates. This allows the large error performance for both broadband and narrowband signal spectra to be handled in a unified manner. Here, however, we restrict our attention to a comparison of our model with theoretical and experimental results

ACKNOWLEDGEMENT This research was supported in part by the Office of Naval Research under contract N00014-81-K-0146.

already reported in the literature for very large BT and broadband lowpass signal spectra.

II. PROBLEM STATEMENT

We briefly outline a common version of the two sensor passive time delay estimation problem. Available are the outputs of two sensors, $x_1(t)$ and $x_2(t)$, over an interval of time $t \in [0, T]$

$$x_1(t) = s(t) + n_1(t)$$
$$x_2(t) = s(t - D) + n_2(t) \tag{1}$$

The signal components, $s(t)$ and a delayed version $s(t - D)$, and the noises, $n_1(t)$ and $n_2(t)$, are ergodic, zero mean, stationary Gaussian random processes posessing spectral densities $G_{ss}(w)$, $G_{n1}(w)$ and $G_{n2}(w)$ respectively. In addition D is restricted to an interval $[-D_m, D_m]$, some *a priori* region of delay. The signal auto-correlation, $R_{ss}(\tau) = \mathbf{E}[s(t)s(t-\tau)]$, is assumed to be essentially zero for $|\tau| > T_c$, the correlation time of the signal, which is also the independence time for this case. This implies that the auto and cross-correlation functions of the observations, $R_{ij}(\tau) = \mathbf{E}[x_i(t)x_j(t-\tau)]$, are non-zero over at most a distance T_c from the time origin, $\tau = 0$.

The correlation method of time delay estimation entails estimation of $R_{12}(\tau)$ by a time average of the product $x_1(t)$ and $x_2(t-\tau)$, denoted $\hat{R}_{12}(\tau)$, the sample cross-correlation function. The cross-correlation estimate of time delay, $\hat{D}$, is the location in time, within $[-D_m, D_m]$, at which the absolute maximum of the sample cross-correlation function occurs. By ergodicity the sample cross-correlation, $\hat{R}_{12}(\tau)$, converges to the true cross-correlation, $R_{12}(\tau) = R_{ss}(\tau - D)$, in the limit of large observation time, T. For finite T, however, $\hat{R}_{12}(\tau)$ may not have its absolute maximum at D.

The local variance of the cross-correlation estimate is derived in [6] as a result of linearizing the first derivative of $\mathbf{E}[R_{12}(\tau)]$ in the neighborhood of D. Consequently the analysis is exact as long as errors are concentrated over the region where $R_{ss}(\tau)$ does not deviate from an inverted parabola centered about D with maximum $R_{ss}(D)$ and curvature $\dfrac{d^2 R_{ss}(\tau)}{d\tau^2}\Big|_{\tau=D}$. This parabola is commonly referred to as the "osculating parabola". Therefore, if the probability that the error falls outside this so called linear region is high, the local variance will give an overly optimistic indication of the performance of the correlator. In the sequel we will refer to the "linear region", mentioned above, as the "small error region", and the remainder of the *a priori* interval as the "large error region" (Fig. 2.1).

III. A LEVEL CROSSING INTERPRETATION FOR LARGE ERROR

In this section a model for the occurence of large errors, or peak ambiguity, is developed. By definition a peak ambiguity occurs when the global peak detection scheme indicates that the absolute maximum of $\hat{R}_{12}(\tau)$ resides outside of the small error region $[D-\delta, D+\delta]$, where δ is the distance from D of the zero associated with the osculating parabola (Fig. 2.1). As an approximation peak ambiguity will be characterized by a conditional exceedance of $\hat{R}_{12}(D)$ by $\hat{R}_{12}(\tau)$ outside of the local region (Fig. 3.1). Define the "ambiguity process" $\Delta R(\tau)$ as the difference $\hat{R}_{12}(\tau) - \hat{R}_{12}(D)$. Then a large error is characterized by the event

$$\Delta R(\tau) > 0 \quad , \quad some\ \tau \in \mathbf{H}_\delta \tag{2}$$

$$\mathbf{H}_\delta = \{t : t \in [-D_m, D-\delta\,] \cup [D+\delta, D_m\,]\}$$

In what follows we will assume a.s. continuity of the ambiguity process in its time argument. Define the occurence of peak ambiguities over some set of intervals Θ contained in $\mathbf{H}_\delta$ by $A_\Theta = \{\Delta R(\tau) > 0 \ , \ some\ \tau \in \Theta\}$. In chapter 5 of [4] it is shown that, as a result of the invariance to $\tau - D$ of the statistics of $\hat{R}_{12}(\tau)$ for $\tau \in \mathbf{H}_\delta$, the actual delay D can be taken as zero. From [4], for lowpass signals, the probability of getting a large error over the *a priori* region, P_e, is given by

$$P_e = P(A_{I_L})[2 - P(A_{I_L})] \tag{3}$$

where

$$I_L = [-D_m, -\delta] \tag{4}$$

If we assign a direction of pursuit to the trajectory of ΔR over the interval I_L then an equivalence can be drawn between peak ambiguities and zero crossings by ΔR over $\mathbf{H}_\delta$. We will let $\Delta R(\tau)$ trace out a path as its time argument increases from $-D_m$ to $-\delta$. Define the cumulative distribution of $\Delta R(\tau)$ at $\tau = t$

$$F(a;\, t) = P(\Delta R(t) \le a) \tag{5}$$

Then the probability that an ambiguity occurs in I_L can be written

$$\begin{aligned} P(A_{I_L}) = {} & P(A_{I_L} | \Delta R(-D_m) \le 0)\ F(\,0;\, -D_m) \\ & + P(A_{I_L} | \Delta R(-D_m) > 0)\ (1 - F(\,0;\, -D_m)) \end{aligned} \tag{6}$$

Where $P(A|B)$ denotes the conditional probability of event A given event B.

A simplification of Eq. (6) is immediate. The conditional probability $P(A_{I_L} | \Delta R(-D_m) > 0)$ is equal to one by definition. Rewrite Eq. (6) with the above fact

$$P(A_{I_L}) = P(A_{I_L} | \Delta R(-D_m) \le 0) F(\,0;\, -D_m) + 1 - F(\,0;\, -D_m) \tag{7}$$

Define the upcrossing count process associated with ΔR

$$N(J) = \left\{ \begin{array}{c} number\ of\ times\ \Delta R\ upcrosses \\ 0\ in\ an\ interval\ J \in H_\delta \end{array} \right. \tag{8}$$

Given $\Delta R(-D_m) < 0$, $N(I_L)$ must be greater than zero for an ambiguity to have occured in I_L. Since the converse is also true we can write

$$P(A_{I_L}) = P_c(N(I_L) > 0)\ F(\,0;\, -D_m) + 1 - F(\,0;\, -D_m) \tag{9}$$

where $P_c(B) = P(B \mid \Delta R(-D_m) \le 0)$.

Using Eq. (9) in Eq. (3) gives an explicit relation between upcrossing probabilities and peak ambiguity probabilities. Thus the determination of the probability of large error is reduced to solving for a level crossing probability. Unfortunately the distribution of level crossings is unknown explicitly except for certain classes of Markov and pseudo-Markov processes [9]. However, motivated by results on the asymptotic distributions of level crossing probabilities in chapter 4 of [4], an approximation to $P_c(N(I) > 0)$ can be applied, based on an inhomogeneous Poisson model. Specifically, the probability of getting k upcrossings in

an interval $[t_o, t]$ is approximated as

$$P(N(t) = k) = \frac{1}{k!}\left[\int_{t_o}^{t}\lambda_c(\tau)d\tau\right]^k \exp\{-\int_{t_o}^{t}\lambda_c(\tau)d\tau\} \tag{10}$$

The integral in (10), $\int_{t_o}^{t}\lambda_c(\tau)d\tau = \mathbf{E}[N(t)|\Delta R(-D_m) > 0)]$, is the conditional rate function of the upcrossings on $[t_o, t]$ and the integrand λ_c is the conditional intensity. For notational simplicity we define the number of zero upcrossings by ΔR over I_L:

$$\mathbf{E}_c[N(I_L)] = \int_{-D_m}^{-\delta}\lambda_c(\tau)d\tau \tag{11}$$

From (10) the Poisson model gives the simple relation

$$P_c(N(I_L) > 0) = 1 - \exp\{-\int_{-D_m}^{-\delta}\lambda_c(\tau)d\tau\} \tag{12}$$

In addition to yielding approximations to the peak ambiguity probability, the Poisson characterization allows a simple expression for the global variance of the time delay estimate to be derived [4, chapter 5],

$$var(\hat{D}) = var_L(\hat{D})\,[1 - P_e] + \int_{-D_m}^{-\delta}\tau^2\,\hat{\lambda}_c(\tau)d\tau \bullet P_e \tag{13}$$

where we have defined the normalized intensity function

$$\hat{\lambda}_c(\tau) = \frac{\lambda_c(\tau)}{\int_{-D_m}^{-\delta}\lambda_c(s)ds} \tag{14}$$

and var_L is the local variance, given by [6]

$$var_L(\hat{D}) = 2\pi\,\frac{\int_{-\infty}^{\infty} w^2 G_{11}(w)G_{22}(w)(1 - |\gamma_{12}(w)|^2)dw}{(\int_{-\infty}^{\infty} w^2 G_{ss}(w)dw)^2} \tag{15}$$

In the above G_{11} and G_{22} are the spectra of the observations x_1 and x_2 respectively.

Eq. (13) shows that, as the probability of large error increases, one must correct the local variance, var_L, by adding a proportionate amount of a potentially large quantity. This quantity can be interpreted as the second moment of the density $\hat{\lambda}_c$ of the occurrence times of ambiguities, i.e. the mean-square deviation of the magnitude of a large error.

IV. EXPLICIT FORM OF THE CONDITIONAL RATE FUNCTION

Of interest here is the form of the conditional rate function $\mathbf{E}_c[N]$, or the time varying intensity function, $\lambda_c(\tau)$, of the number of upcrossings N

associated with the ambiguity process ΔR. λ_c is a function of the bivariate distributions of ΔR conditioned on $\Delta R(-D_m) < 0$. The bivariate distribution of the sample cross-correlation function, $\hat{R}_{12}$, is not exactly known except for particularly simple cases such as sinusoidal signals [10]. Even in this simple situation the resultant expressions are cumbersome. Here we will assume that R_{12} is adequately characterized by Gaussian statistics. This can be justified if one has a large observation interval and implements segment averaging techniques to estimate the cross-correlation function [4, chapter 5]. It is shown in [4] that if $\hat{R}_{12}$ satisfies some regularity conditions, involving the continuity of certain first order derivatives of its covariance function and a non-degeneracy condition, then the conditional intensity of the zero upcrossings is given by

$$\lambda_c(\tau) = K \int_0^\infty y\ \Phi(\alpha y + \vartheta)\ \varphi(y + \Theta) dy \quad , \qquad \tau \in [-D_m, -\delta\ ] \tag{16}$$

In (16) K, α, ϑ and Θ are functions of τ given in [4, chapter 5]. The functions Φ and φ are the standard Gaussian distribution and density respectively.

For purposes of comparison, it will be instructive to consider Eq. (16) under similar broadband assumptions to those in [3]. For large $D_m B$, and flat lowpass signal and noise spectra, it can be shown [4] that the intensity (16) is independent of time. Therefore the conditional rate function of the upcrossings (the integral of (16)) takes the form

(17)

$$\mathbf{E}_c[N] = \lambda_c(D_m - \delta) = \frac{B_o(D_m - \delta)}{\sqrt{3}\ \sqrt{2 + |\gamma_{12}|^2}} \frac{\Phi\left[\dfrac{\gamma' \sqrt{2B_o T}}{\sqrt{3 + 2|\gamma_{12}|^2}}\right]}{\Phi(\gamma' \sqrt{2B_o T})} \varphi(\gamma' \sqrt{2B_o T})$$

Here B_o is the single sided bandwidth of the signal spectrum in Hertz, $|\gamma_{12}|$ is the magnitude coherency between x_1 and x_2

$$|\gamma_{12}| = \frac{G_{ss}}{(G_{ss} + G_{n1})(G_{ss} + G_{n2})} \tag{18}$$

and we have defined

$$\gamma' = \frac{|\gamma_{12}|}{\sqrt{2 + |\gamma_{12}|^2}} \tag{19}$$

The consideration of the above gives some insight into the large error performance of the correlator. First, consistent with the assumptions invoked by Ianniello in [2] for large BT, the time independence of λ_c suggests that erroneous estimates are uniformly distributed over the *a priori* interval. From (17) we note that, as expected, decreasing the SNR, or equivalently decreasing $|\gamma_{12}|$ and γ' towards zero, increases the rate of errors. However, the rate of increase of the rate function (17) as the SNR decreases from infinity is determined by the time-bandwidth product $B_o T$ and the *a priori*-interval-bandwidth product $B_o D_m$. Specifically, for fixed D_m, $B_o T$ determines the signal-to-noise ratio threshold beyond which increasing SNR leads to a rapid decrease in λ_c. In general, increasing $B_o T$ shifts this threshold to a lower SNR, expanding the range of error free operation of the correlator. On the other hand, $B_o D_m$ affects the

magnitude of the expected number of upcrossings, and its rate of increase, as the SNR decreases beyond the threshold. Increasing the length of the *a priori* region, $[-D_m, D_m]$, naturally makes large errors more likely as is indicated by the corresponding increase in $\mathbf{E}_c[N]$ of (17). The above comments corroborate analytical results obtained in [5] and the generally acknowledged practical performance of the correlator as a time delay estimator [2].

Consider the asymptote of $\mathbf{E}_c[N]$ as the SNR goes to infinity. In this case $|\gamma_{12}|$ goes to one and , from Eq (17)

$$\mathbf{E}_c[N] \rightarrow \frac{B_o(D_m - \delta)}{3} \frac{\Phi(k_1\sqrt{B_o T})}{\Phi(k_2\sqrt{B_o T})} \varphi(k_2\sqrt{B_o T}) \tag{20}$$

where k_1 and k_2 are finite numerical constants. This interesting result implies that even at infinite SNR large errors may be committed. This is in agreement with results obtained by Ianniello in [7] and [8] showing the sub-optimality of the cross-correlation estimate of delay for finite BT.

V. NUMERICAL COMPARISONS

We conclude with a comparison of the Poisson motivated global variance approximation (13) with the experimental variance of the cross-correlator for broadband signals and large BT reported in a simulation study by Scarbrough *et al* [3]. The time independent intensity, used to obtain expression (17), was substituted into the Poisson approximation to the conditional probability of large error (12) which was in turn used in P_e (3) to obtain the simple form of the global variance approximation (13)

$$var(\hat{D}) = var_L(\hat{D})(1 - P_e) + \frac{D_m^2}{3} P_e \tag{21}$$

where

$$P_e = 1 - e^{-2\lambda_c D_m} \Phi^2(\gamma'\sqrt{2B_o T}) \tag{22}$$

and

$$var_L(\hat{D}) = \frac{3}{8\pi^2} \frac{1 - |\gamma_{12}|^2}{B_o^3|\gamma_{12}|^2} \tag{23}$$

Here λ_c is the time independent intensity and δ is neglected in (21) due to its small magnitude with respect to $-D_m$ for large BT. In the sequel we simply refer to the implementation of the global variance approximation (21) as the Poisson approximation.

In [3] the cross-correlator was simulated for signals and noises with flat lowpass spectra of bandwidth $B_o = 100Hz$. The observation time was set to $T = 8secs.$ giving a value BT near 1000 and the *a priori* maximum absolute deviation of delay was set to 1/8 second. Fig. 5.1 shows a plot of the Poisson approximation as a function of the input SNR at a single sensor. Also displayed are plots of the local variance approximation var_L (15) (CRLB in figure), the global variance approximation of Ianniello, and the simulated error performance of the cross-correlation delay estimate [3] (indicated by "o" on Fig. 5.1). As compared to the simulation, all of the above variance approximations are accurate above a SNR of about -6dB. However, as the SNR falls below this threshold, the Poisson and Ianniello's approximations agree well with the simulation results in contrast

to the local variance. This is the so called ambiguity dominated region, where large errors dominate the performance of the correlator. Recall, however, that Ianniello's approximation involves the numerical evaluation of a complicated integral while the Poisson approximation (21) is in explicit form.

VI. CONCLUSION

A general framework for large errors in correlation based time delay estimation was developed in terms of level crossing probabilities. This allows the use of the rate function of the large errors as an indicator of performance for arbitrary signal spectra and observation times. A Poisson approximation to the distribution of the number of level crossings, or peak ambiguities, was invoked to derive a simple expression for the global variance of the correlation estimate of time delay. A general expression for the expected number of ambiguities (upcrossings of the ambiguity process) was derived under a Gaussian model for the cross-correlation.

Analysis of the intensity of the large errors for lowpass signal and noise spectra and high BT reveals that the errors are equally likely to occur throughout the *a priori* region. This is supportive of the conjectured uniformity property which was neccessary for the derivation of Ianniello's approximation in [2]. Under the above observation model, the expression governing the average number of ambiguities over a large *a priori* region of delay reduced to a particularly simple form. The resulting expression behaves in a manner consistent with previous studies of the performance trade-offs that exist between SNR and time- bandwidth product undertaken in [2], [3], [5] [7] and [8]. Not so explicitly identified elsewhere is the role of the *a priori*-interval-bandwidth product in determining performance. Indeed while the time-bandwidth product determines the location of the SNR threshold, beyond which large errors predominate estimator error, the *a priori*-interval-bandwidth product controls the rapidity with which the performance degenerates beyond the threshold.

In this paper only the large BT, lowpass signal situation was considered. More general results are available [4] for arbitrary BT and signal spectra which show some interesting differences between the simple case considered here and the case of narrow bandpass spectra. Although no simulation results are available for validation, the bandpass results are consistent with the behavior of the Ziv-Zakai lower bound on estimator variance reported by Weiss *et al* in [5]. Combined with the results reported here, this lends strong support to the application of level crossing statistics in characterizing the performance of correlation type time delay estimates. Possible extensions of this approach to large deviations in general maximum likelihood estimates are currently being explored.

REFERENCES

1. G.C. Carter, "Time delay estimation for passive sonar signal processing," *IEEE Trans. on ASSP*, Vol. ASSP-29, No. 3, pp. 463-470, 1981.
2. J.P. Ianniello, "Time delay estimation via cross-correlation in the presence of large estimation errors," *IEEE Trans. on ASSP*, Vol. ASSP-30, No. 6, pp. 998-1003, 1982.

3. K. Scarbrough, R.J. Tremblay, and G.C. Carter, "Performance predictions for coherent and incoherent processing techniques of time delay estimation," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-31, No. 5, pp. 1191-1196, 1983.

4. A. Hero, *Topics in time delay estimation* Ph.D dissertation, Princeton University, Princeton, N.J. 08544, 1984.

5. A. Weiss and E. Weinstein, "Fundamental limitations in passive time delay estimation - part I: narrowband systems" *IEEE Trans. Acoust., Speech, Signal Proc.*, Vol. ASSP-31, No. 2, pp. 472-485, 1983.

6. V.H. MacDonald and P.M. Schultheiss, "Optimum passive bearing estimation in a spatially incoherent noise environment", *Journal of the Acoustical Society of America*, Vol. 46, No. 1, Part 1, pp. 37-43, 1969.

7. J.P. Ianniello, E. Weinstein, A. Weiss, "Comparison of the Ziv-Zakai lower bound on time delay estimation with correlator performance," *ICASSP-83 Proceedings*, Vol. 2 , pp. 875-878.

8. J.P. Ianniello, "A discussion of the maximum likelihood estimator for two channel time delay estimation", Technical Memorandum, Naval Underwater Systems Center, New London Lab, New London, CT 06320, April 1984.

9. J. Abrahams, "A survey of recent progress on level crossing problems for random processes," *Office of Naval Research*, Tecnical Memorandum, June 21, 1983.

10. L.E. Miller and J.S. Lee, "Bandpass correlator analysis for general input assumptions," *IEEE Trans. Inform. Theory*, Vol. IT-28, No. 6, pp. 973-976, 1982.

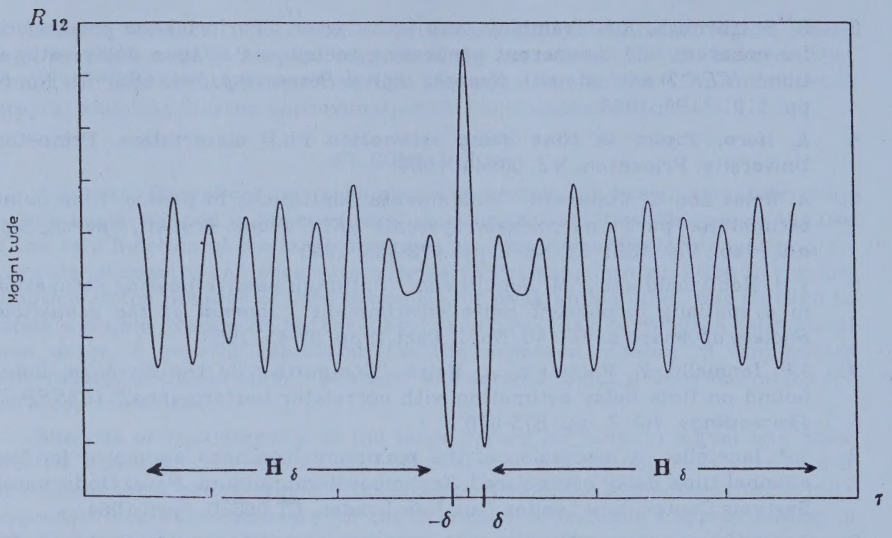


Fig. 2.1

The large error region, H_δ, is determined by the zeros of the osculating parabola. In the figure $D = 0$.

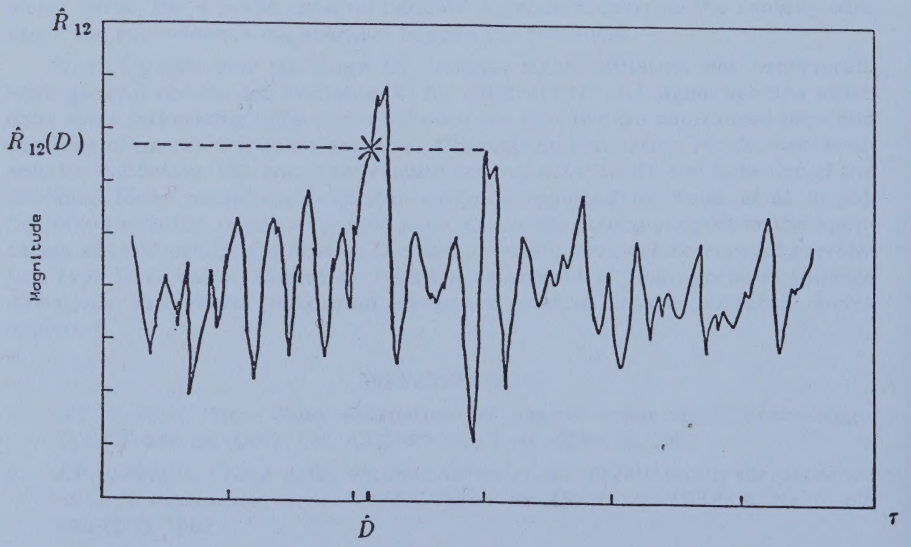


Fig. 3.1

A large error occurs when $\hat{R}_{12}(\tau)$ exceeds $\hat{R}_{12}(D)$ somewhere within the large error region. This is modeled by the upcrossing labeled "X" in the figure.

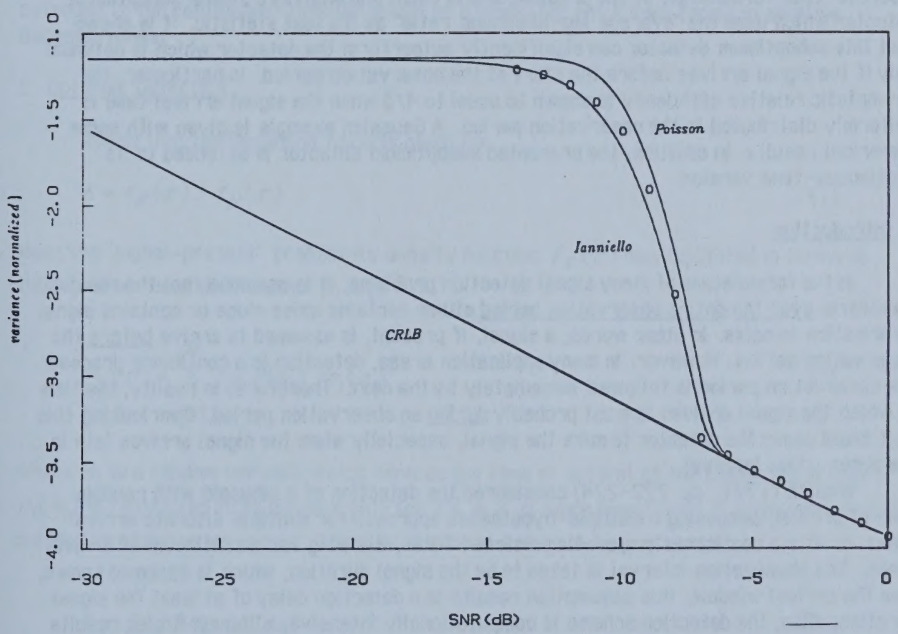


Fig. 4.1
The Poisson and Ianniello's variance approximations are in good agreement with the observed variance obtained by experiment ("o" in the figure), while the local variance (CRLB) significantly underestimates the actual variance.

DETECTION OF SIGNALS WITH RANDOM ARRIVAL TIME

L.A. LONGLEY and C.C. LEE
Dept. of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60201

L.A. Longley also with
Data Systems Research Lab
Motorola Inc.
Schaumburg, IL 60196

ABSTRACT

The detection of signals with random arrival time is considered. We propose a discrete-time formulation of the problem, and present and analyze a simple suboptimum detector which uses the "average log-likelihood ratio" as the test statistic. It is shown that this suboptimum detector can significantly outperform the detector which is optimum only if the signal arrives before the start of the observation period. In particular, the asymptotic relative efficiency is shown to equal to 4/3 when the signal arrival time is uniformly distributed in the observation period. A Gaussian example is given with some numerical results. In addition, the presented suboptimum detector is extended to its continuous-time version.

1. Introduction

In the formulation of many signal detection problems, it is assumed that the received waveform over the entire observation period either contains noise alone or contains signal information in noise. In other words, a signal, if present, is assumed to arrive before the observation period. However, in many application areas, detection is a continuing process: one observation period is followed immediately by the next. Therefore, in reality, the time at which the signal arrives is most probably during an observation period. Overlooking this fact could cause the detector to miss the signal, especially when the signal arrives late in the observation interval.

Whalen (1971; pp. 222-224) considered the detection of a sinusoid with random time of arrival, proposing a multiple-hypotheses approach for multiple discrete arrival times, or else a continuously operating matched filter, allowing for a continuum of arrival times. The observation interval is taken to be the signal duration, which is assumed known, plus the arrival window; this assumption results in a detection delay of at least the signal duration. Also, the detection scheme is computationally intensive, although it also results in an estimate of the arrival time. In addition, no performance analysis is provided. Helstrom (1975; pp. 174-180, 295-298) studied the detection of a quasiharmonic signal in noise, but he assumed the observation interval to be much longer than the signal duration, thus greatly extending the detection delay. His approach also results in a complex implementation.

In this paper, we consider the detection of a signal with random arrival time in stationary white noise, and attempt to minimize the detection delay and computational complexity. Specifically, we formulate the problem in a discrete-time manner as follows:

$$K\colon\ r(n) \sim f_K(r) \qquad 1 \le n \le N$$

versus

$$H\colon\ r(n) \sim f_H(r) \qquad 1 \le n \le N$$

where f_K and f_H represent the joint probability densities under noise-alone and signal-present conditions, respectively, $\{r(n)\}$ are independent samples from the received waveform, and $n \in [1, N]$ is the observation interval. Since the scheme is oriented toward detecting the signal arrival, it is assumed that if the signal does arrive

during the observation interval (hypothesis K), then it is present for the remainder of that period. Otherwise, no assumptions are made about the signal duration.

We derive in Sec. 2 the optimum detector for this problem. Because the optimum detector is very difficult to analyze and is relatively difficult to implement, we present in Sec. 3 a suboptimum alternative which uses the "average log-likelihood ratio" as the test statistic instead of the "average likelihood ratio". We compare its performance to the detector which is optimum only if the signal arrives before the start of the observation period. The asymptotic relative efficiency is derived and is computed for a uniformly distributed signal arrival time. The presented suboptimum detector is as easy to implement as the referenced detector and can be significantly more efficient. Then, an example concerning the detection of a constant signal in white Gaussian noise is given with some numerical results. Finally, we show that the presented suboptimum detector can be extended to the continuous-time detection of a signal (of any known form) in additive Gaussian noise.

2. Optimum Detection

Recall that the likelihood ratio is given by

$$\Lambda = f_K(\boldsymbol{r}) \,/\, f_H(\boldsymbol{r}) \tag{1}$$

Now, the "signal-present" probability density function $f_K(\boldsymbol{r})$ may be stated in terms of the signal-arrival distribution $p_1(n_1)$:

$$f_K(\boldsymbol{r}) = \sum_{n_1=1}^{N} f_K(\boldsymbol{r}|n_1) \cdot p_1(n_1) \tag{2}$$

where n_1 is a random variable which denotes the time of arrival of the signal. Note that $p_1(n_1)$ is restricted to be nonzero only for $1 \le n_1 \le N$, since hypothesis K presumes arrival during the observation period. Then the likelihood ratio can be restated as

$$\Lambda = \sum_{n_1=1}^{N} \Lambda(n_1) \cdot p_1(n_1) \tag{3}$$

in terms of the conditional likelihood ratio

$$\Lambda(n_1) = f_K(\boldsymbol{r}|n_1) \,/\, f_H(\boldsymbol{r}) \tag{4}$$

The likelihood ratio Λ is thus the average, with respect to n_1, of the conditional likelihood ratios $\Lambda(n_1)$. Now, the conditional likelihood ratio can be stated in terms of the individual time samples:

$$\Lambda(n_1) = \prod_{n=n_1}^{N} \frac{g_K[r(n)|n_1]}{g_H[r(n)]} \tag{5}$$

where g_K and g_H represent the probability density functions for a single sample $r(n)$,

under hypotheses K and H, respectively. Therefore, the Neyman-Pearson optimum detector is

$$\Lambda = \sum_{n_1=1}^{N} p_1(n_1) \prod_{n=n_1}^{N} \gamma(n,n_1) \quad \begin{matrix} > \\ \Lambda_0 \\ < \end{matrix} \quad \begin{matrix} \rightarrow K \\ \\ \rightarrow H \end{matrix} \tag{6}$$

where

$$\gamma(n,n_1) \equiv g_K[r(n)|n_1] \,/\, g_H[r(n)] \tag{7}$$

and Λ_0 is the decision threshold. Since the signal itself is fixed, and only its arrival time n_1 is undetermined, $\gamma(n,n_1)$ depends only on the time <u>difference</u> $n-n_1$, so that (6) may be specified in terms of "$\gamma(n-n_1+1)$":

$$\Lambda = \gamma(1)\,[\gamma(2)\ldots\,[\gamma(N)p_1(1) + p_1(2)\,]\ldots + p_1(N)\,] \quad \begin{matrix} > \\ < \Lambda_0 \\ \end{matrix} \quad \begin{matrix} \rightarrow K \\ \rightarrow H \end{matrix} \tag{8}$$

This may be implemented recursively as

$$z(n) = \gamma(n)\,[z(n+1) + p_1(N-n+1)\,] \qquad n = 1, 2, \ldots, N-1 \tag{9a}$$
$$= \gamma(N)\,p_1(1) \qquad n = N$$

with the detection decision

$$\Lambda = z(1) \quad \begin{matrix} > \\ < \Lambda_0 \end{matrix} \quad \begin{matrix} \rightarrow K \\ \rightarrow H \end{matrix} \tag{9b}$$

As a special case, if the signal arrives at the beginning of the observation period, the optimum detector is much simpler:

$$\Lambda = \Lambda(1)$$

or

$$\Lambda = \prod_{n=1}^{N} \gamma(n) \quad \begin{matrix} > \\ \Lambda_0 \\ < \end{matrix} \quad \begin{matrix} \rightarrow K \\ \\ \rightarrow H \end{matrix} \tag{10}$$

This detector may be readily converted to an even simpler summation by taking the logarithm of the likelihood ratio. However, in comparison with this simple case, the more general optimum detector (9) is relatively difficult to implement and, above all, very difficult to analyze. In the next two sections, we propose and analyze a suboptimum detector which can significantly outperform the detector (10) when the signal arrival time is random.

3. Average Log-Likelihood Detection

Since the latter optimum detector may be cumbersome to compute, a suboptimum detector is also be considered. Now, the conditional likelihood ratio $\Lambda(n_1)$ may be considerably simplified by taking the logarithm, which converts the product (5) into the following summation:

$$\mathcal{L}(n_1) = \log \Lambda(n_1)$$

$$= \sum_{n=n_1}^{N} \log \frac{g_K[r(n)\,|\,n_1]}{g_H[r(n)]} \tag{11a}$$

Then, the "average log-likelihood detector" (ALLD) is proposed:

$$L_A = \sum_{n_1=1}^{N} \mathcal{L}(n_1) \cdot p_1(n_1) \;\underset{H}{\overset{K}{\gtrless}}\; T_0 \tag{11b}$$

where T_0 is the decision threshold.

While not an optimum detector, its simplicity makes the ALLD useful in certain applications. For example, for an arbitrary signal $s(n)$ arriving at time n_1, in additive noise $n(\cdot)$, the alternative hypotheses are

$$K:\ r(n) = n(n) \qquad 1 \le n < n_1$$
$$= n(n) + s(n-n_1+1) \qquad n_1 \le n \le N$$

or

$$H:\ r(n) = n(n) \qquad 1 \le n \le N$$

where $s(n)$ is defined for $1 \le n \le N$ and the noise $n(\cdot)$ has variance σ^2. When $n(\cdot)$ is a zero-mean, white Gaussian process, and the arrival distribution is the least favorable one, i.e. uniform

$$p_1(n_1) = 1/N \qquad 1 \le n_1 \le N \tag{13}$$
$$= 0 \qquad n_1 \text{ otherwise}$$

the ALLD becomes a correlation receiver

$$L_A = (1/\sigma^2) \sum_{n=1}^{N} r(n)\, S(n) \;\underset{H}{\overset{K}{\gtrless}}\; T_0 \tag{13a}$$

using the "cumulative signal sum":

$$S(n) \equiv (1/N) \sum_{n_1=1}^{n} s(n_1) \tag{13b}$$

4. Detection of a Constant Signal and Performance Analysis

In the special case where the signal is a constant s, the ALLD (13) is given by

$$L_R \equiv (s/\sigma^2) \sum_{n=1}^{N} r(n)(n/N) \underset{H}{\overset{K}{\gtrless}} T_0 \qquad (14a)$$

This detector has the advantages of good performance, intuitive appeal, and mathematical tractability. Because of the (n/N) weighting factor, the received waveform later in the observation block is relatively "more important" in the detection decision at time $n=N$, so that the probability of detecting "late arrivals" is improved. Also, note that the following conventional linear detector (LD) is optimum for detecting a constant signal s in Gaussian noise, when the arrival time is at (or before) the start of the observation period:

$$L_L \equiv (s/\sigma^2) \sum_{n=1}^{N} r(n) \underset{H}{\overset{K}{\gtrless}} T_0 \qquad (14b)$$

We call (14a) the "linear-ramp detector" (LRD) after comparision to the linear detector (14b).

For performance analysis regarding a constant signal s, the LRD and LD are redefined in terms best suited for analysis:

$$L_R \equiv (\sqrt{3}/N^2) \sum_{n=1}^{N} n\, r(n) \underset{H}{\overset{K}{\gtrless}} T_0 \qquad (15a)$$

and

$$L_L \equiv (1/N) \sum_{n=1}^{N} r(n) \underset{H}{\overset{K}{\gtrless}} T_0 \qquad (15b)$$

For $n(\cdot)$ a zero-mean, white noise process, not necessarily Gaussian, with variance σ^2, it is apparent that under hypothesis H, both L_R and L_L are zero-mean random variables. Also, it is readily shown that variance of both quantities is σ^2/N, under H (and K). Furthermore, since the time samples $r(n)$ are independent random variables with noninfinitesimal variance, their CDFs are essentially Gaussian for $N \gg 1$, according to the Central Limit Theorem (Papoulis, 1965; pp. 266-268). Now, the false-alarm probability α (i.e., the probability of deciding K, under H) is equal to the probability that the detector output L_R (or L_L) exceeds the decision threshold T_0. Then for both the LRD and the LD,

$$\alpha = Q[T_0/(\sigma/\sqrt{N})] \qquad (16a)$$

where $Q(\cdot)$ is one minus the standardized Gaussian CDF. When α is prespecified, (16a) may be solved to determine the decision threshold:

$$T_0 = (\sigma/\sqrt{N})\, Q^{-1}(\alpha) \qquad (16b)$$

Next, under hypothesis K, the following expected values of L_R and L_L, given the signal arrival time n_1, are obtained:

$$E_K[L_R|n_1] = (\sqrt{3}/2)\,[(N^2+N-n_1^2+n_1)/N^2]\,s \qquad (17a)$$

and

$$E_K[L_L|n_1] = (n_1/N)\,s \qquad (17b)$$

The conditional detection probability $\beta(n_1)$ (i.e., the probability of correctly deciding K, given n_1) may then be stated, where $E_K[L|n_1]$ below is replaced by the appropriate expression from (17a) or (17b):

$$\beta(n_1) = Q\{(T_0 - E_K[L|n_1])/(\sigma/\sqrt{N})\} \qquad (18)$$

Finally, the overall detection probability β is

$$\beta = \sum_{n_1=1}^{N} \beta(n_1)\,p_1(n_1) \qquad (19)$$

Based on the above analysis, the discrete-time LRD (15a) has been compared to the LD (15b) over a range of false-alarm probabilities (α) and signal-to-noise ratios, for a constant signal with a uniform arrival distribution, in white noise. As seen in Figure 1, the LRD exhibits superior performance in all cases, in terms of higher probability of detection (β), for a given signal-to-noise ratio ($10\cdot\log_{10}E[s^2/\sigma^2]$). In particular, the left column of Figure 1 (i.e., (a), (c), and (e)) shows results for N = 100 samples, while the right column of Figure 1 ((b), (d), and (f)) shows results for N = 1000 samples. Results for α = 0.1, α = 0.01, and α = 0.001, are seen in the first row (i.e., (a) and (b)), second row ((c) and (d)), and third row ((e) and (f)), respectively. For each case, the improvement diminishes for β above 90 %.

The latter analysis also forms the basis for an "asymptotic relative efficiency" (ARE) (Capon, 1961) comparison between the two detectors, an indication of performance under low signal-to-noise conditions. First, the overall expected value of the test statistic under hypothesis K must be obtained according to

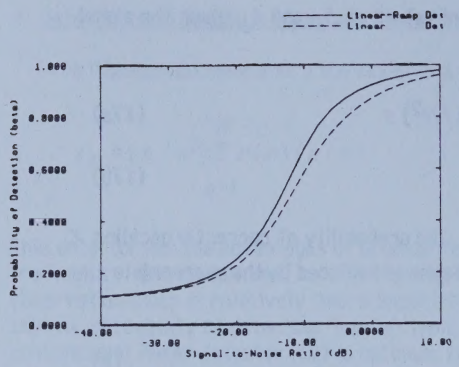
$$E_K[L] = \sum_{n_1=1}^{N} E_K[L|n_1]\,p_1(n_1) \qquad (20)$$

Then substituting from (17) into (20) yields

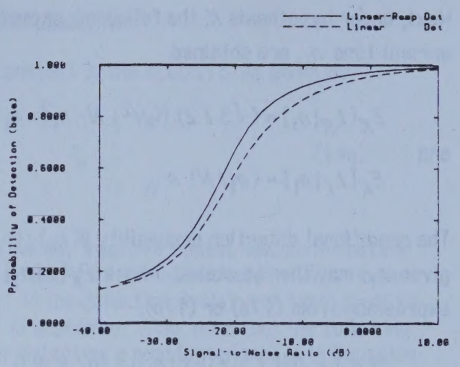
$$E_K[L_R] = (\sqrt{3}/3)\,s\,[1 + 3/(2N) + 1/(2N^2)] \qquad (21a)$$

and

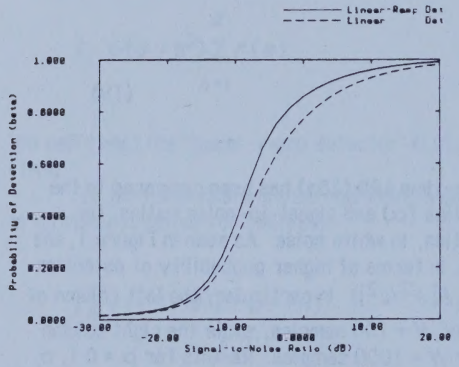
$$E_K[L_L] = (1/2)\,s\,(1 + 1/N) \qquad (21b)$$



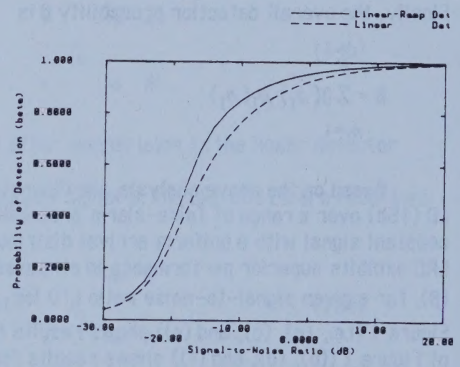
(a) N = 100 samples, α = 0.1



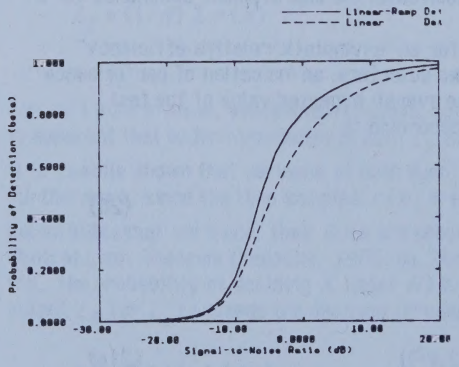
(b) N = 1000 samples, α = 0.1



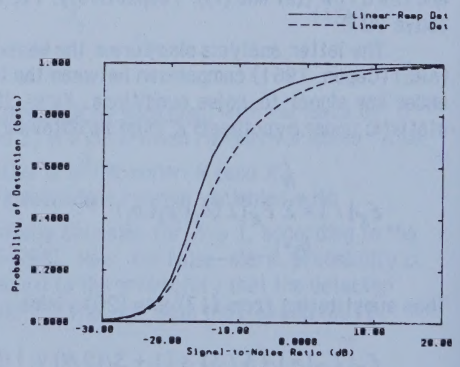
(c) N = 100 samples, α = 0.01



(d) N = 1000 samples, α = 0.01



(e) N = 100 samples, α = 0.001



(f) N = 1000 samples, α = 0.001

Figure 1 Probability of Detection (β) for a constant signal with a uniform arrival distribution, versus Signal-to Noise ratio in dB, in white noise. The solid and dashed curves correspond to the Linear-Ramp Detector and Linear Detector, respectively. False-alarm probability is denoted by α.

The ARE is the ratio of the "efficacy" of each detector, computed as follows:

$$\mathfrak{E} = \lim_{N \to \infty} \left(d/ds \{E_K[L]\}|_{s=0}\right)^2 / N \operatorname{var}_H[L] \qquad (22)$$

resulting in

$$\mathfrak{E}_R = 1 / 3\sigma^2$$

and

$$\mathfrak{E}_L = 1 / 4\sigma^2$$

Finally, the ARE is obtained:

$$ARE_{R,L} = 4 / 3$$

This result indicates that in low signal-to-noise conditions, the LD requires 4/3 as many samples as the LRD, to achieve the same level of performance in detecting a constant signal in additive noise, provided that the signal arrival time is uniformly distributed in $[1, N]$. However, if the signal is known to always arrive at (or before) the beginning of the observation interval, then a similar analysis will verify that the LD is superior:

$$ARE_{R,L} = 3 / 4$$

5. Continuous-Time Formulation

Now we consider the continuous-time formulation. Similar to the discrete-time case, the alternative hypotheses are

$$K: \; r(t) \sim f_K(r) \qquad 0 < t < T$$

versus

$$H: \; r(t) \sim f_H(r) \qquad 0 < t < T$$

where $t \in (0, T)$ is the observation period. Also in direct analogy with the discrete-time formulation, the continuous-time likelihood ratio (23) is a weighted sum of conditional likelihood ratios (24):

$$\Lambda = \int_0^T \Lambda(t_1) \cdot f_1(t_1) \, dt_1 \qquad (23)$$

where

$$\Lambda(t_1) = f_K(r|t_1) / f_H(r) \qquad (24)$$

and $f_1(t_1)$ is the probability density function of the signal-arrival time t_1.

Consider the case of an arbitrary signal $s(t)$, defined for $0 < t < T$, arriving at time t_1, in additive noise $n(t)$:

$$K: \; r(t) = n(t) \qquad 0 < t < t_1$$
$$= n(t) + s(t - t_1) \qquad t_1 \leq t < T$$

or

$$H: \; r(t) = n(t) \qquad 0 < t < T$$

For $n(t)$ a zero-mean, white Gaussian noise process with power spectral density $N_0/2$, the conditional likelihood ratio $\Lambda(t_1)$ (Whalen, 1971; pp. 156-159) can be shown to be

$$\Lambda(t_1) = \exp\left[(1/N_0)\int_{t_1}^{T} [2r(t)\, s(t - t_1) - s^2(t - t_1)]\, dt\right] \tag{25}$$

The Neyman-Pearson optimum detector, <u>for t_1 fixed</u>, is obtained by additively absorbing the term (in the natural log of $\Lambda(t_1)$) that is independent of $r(t)$ into the threshold (which does not affect the optimality of the detection scheme), resulting in the linear detector

$$\Lambda(t_1) = (2/N_0)\int_{t_1}^{T} r(t)\, s(t - t_1)\, dt \;\underset{H}{\overset{K}{\gtrless}}\; T_0 \tag{26}$$

The Neyman-Pearson optimum detector, <u>for t_1 random</u>, uses the likelihood ratio Λ (23) as the test statistic, where Λ is defined using the conditional likelihood ratio $\Lambda(t_1)$ (25). However, no simple calculation can be given for Λ (as was done for $\Lambda(t_1)$ for t_1 fixed, resulting in $\Lambda(t_1)$ (26)), because it involves the averaging, with respect to t_1, of the exponential functions $\Lambda(t_1)$. Therefore, the continuous-time average log-likelihood detector is proposed:

$$\Lambda_A \equiv \int_0^T \log \Lambda(t_1) \cdot f_1(t_1)\, dt_1 \;\underset{H}{\overset{K}{\gtrless}}\; T_0 \tag{27}$$

In many cases, it is reasonable to assume that t_1 is uniformly distributed in the range $0 \leq t_1 \leq T$:

$$f_1(t_1) = 1/T \qquad 0 \leq t_1 \leq T \tag{28}$$
$$= 0 \qquad t_1 \text{ otherwise}$$

The ALLD (27) becomes, after changing the order of integration,

$$\Lambda_A = (2/N_0\, T)\int_0^T r(t) \int_0^t s(t_1)\, dt_1\, dt \;\underset{H}{\overset{K}{\gtrless}}\; T_0$$

or

$$\Lambda_A = (2/N_0)\int_0^T r(t)\, S(t)\, dt \;\underset{H}{\overset{K}{\gtrless}}\; T_0 \tag{29a}$$

where

$$S(t) \equiv (1/T) \int_0^t s(t_1)\, dt_1 \qquad (29b)$$

Thus, under the above assumptions of a uniform arrival distribution and Gaussian noise, the average log-likelihood detector leads to a linear detector. However, for this case, the weighting factor $S(t)$ is the integral of the signal $s(t)$, whose detection is sought. For conventional linear detection, the weighting factor is the signal $s(t)$ itself.

Also, note that for the arrival distribution

$$f_1(t_1) = \delta(t_1 - t_0) \qquad 0 < t_0 < T \qquad (30)$$

which specifies arrival of the signal at time $t_1 = t_0$, or else no arrival during the block, the ALLD (27) degenerates to the conventional linear detector, i.e., $L_A(t_1) = L(t_1)$.

6. Continuous-Time Detection of a Constant Signal

As a special case, consider the detection of a constant (dc) signal s in white Gaussian noise $n(t)$. For this case, the ALLD for a uniform arrival distribution and Gaussian noise (29) simplifies to the continuous-time version of the LRD (cf. (14a)):

$$L_R = (2s/N_0) \int_0^T r(t)(t/T)\, dt \qquad \begin{matrix} > & \rightarrow & K \\ < & T_0 & \rightarrow & H \end{matrix} \qquad (31)$$

However, if the signal is assumed to have arrived at the beginning of or before the observation period, the Neyman-Pearson detector is the well-known linear detector (cf. (14b)):

$$L_L = (2s/N_0) \int_0^T r(t)\, dt \qquad \begin{matrix} > & \rightarrow & K \\ < & T_0 & \rightarrow & H \end{matrix} \qquad (32)$$

This result may be obtained by setting $t_1 = 0$ and $s(t) = s$ in (26).

REFERENCES

Capon, J. "On the Asymptotic Efficiency of Locally Optimum Detectors", IRE Trans. on Info. Theory, vol. IT-7, pp. 67-71, April 1961.

Helstrom, C.W. Statistical Theory of Signal Detection, 2nd ed. Pergamon Press, 1975.

Papoulis, A. Probability, Random Variables, and Stochastic Processes. McGraw-Hill, 1965.

Whalen, A.D. Detection of Signals in Noise. Academic Press, 1971.

ON ADAPTIVE PARAMETER IDENTIFICATION OF LINEAR SYSTEMS WITH APPLICATION TO ADAPTIVE CONTROL/OBSERVERS

by

E. Emre
Dept. of Elec. Eng./Comp. Scie.,
Texas Tech University
Lubbock, TX 79409

0. ABSTRACT

An approach to adaptive parameter identification of linear systems where the parameters enter the I-O map nonlinearly is given. Relevant structure of the parameterized systems is studied as systems over a ring of functions. A technique for adaptive parameter identification of a class of systems is given which can be viewed as a time-varying Newton-like technique. Implications to globally stable adaptive control/observers exist.

1. INTRODUCTION

In this paper, we develop an approach to the problem of nonlinear adaptive parameter identification of (not completely known) parameterized linear systems $\Sigma(\alpha) = (F(\alpha),G(\alpha),H(\alpha))$ where F,G,H are functions of α in some subset Ω of $\mathbb{R}^N$. α is the vector of unknown parameters, and Ω is the parameter space. This is important in two respects. First, it is important to be able to do adaptive parameter estimation from a purely system identification-parameter estimation point of view. Second, these techniques yield some parameter adjustment techniques for use in globally stable adaptive control/observer schemes. First, the structure of parameterized systems from an identification point of view are studied as linear systems over a ring of functions. This study is motivated by, and provides an extension of the approach to linear identification of stable systems in [2]. Then, an adaptive nonlinear asymptotic parameter identification technique is developed for a class of multivariable parameterized systems where the parameter vector α can enter I-O map nonlinearly. These are extensions of some linear techniques used in [1] for a type of implicitly parameterized systems to nonlinear techniques which can be used for a class of multivariable parameterized systems (where the dependence of the transfer matrix coefficients on the unknown parameters are allowed to be twice continuously differentiable functions).

Our technique can be viewed as a time-varying Newton-like technique for solving a class of nonlinear algebraic equations which are time-varying.

This research was supported in part by the National Science Foundation under NSF Research Grant ECS-8204512, in part by the ONR Contracts 76-C-1136 and N00014-84-C-0104, and in part by the Air Force Office of Scientific Research Grant AFOSR-82-0282.

We assume the following conditions on the parameterization of $\Sigma(\alpha)$, which are needed for a globally convergent adaptive identifier with a fixed structure to exist, which identifies the true value of α, α^*, for any value α^* may take in Ω = the parameter space. The first condition is a uniform parameterization (a parameterization with a fixed structure on Ω) condition. This is needed for a single adaptive identifier to be able to estimate α^* wherever $\alpha^* \varepsilon \Omega$ may be. The second condition is extendably identifiability. This is needed here for the adaptive identifier to uniquely determine α^*, wherever it is initialized in $\mathbb{R}^N$, in a globally convergent way. The third condition is that of a "sufficiently exciting input," which is known to be necessary for any kind of system identification technique to uniquely identify a linear system (see e.g., [1-3].)

As it will be clear, our assumptions are also necessary in all previous globally convergent adaptive identification schemes in the literature which utilize a certain type of implicit parameterization. Also, the extendably identifiability condition is automatically satisfied locally for any uniformly parameterized locally identifiable system. Our techniques provide an extension of the existing techniques to explicitly parameterized systems where the functional dependence of $\Sigma(\alpha)$ on α is utilized.

Finally, the results given here can be combined with the results developed in [4] to obtain globally stable adaptive control schemes for a class of parameterized multivariable systems. For this, and for use in adaptive observers, see [4].

2. ADAPTIVE PARAMETER IDENTIFICATION

Throughout this section, we will assume that the entries of $F(\alpha), G(\alpha), H(\alpha)$ are at least twice continuously differentiable on Ω.

2.1 DEFINITION. Let $\Sigma(\alpha)$ be a parameterized system over K. Then, $\Sigma(\alpha)$ is uniformly parameterized iff there exists an M-vector valued function

$$P: \Omega \to R^M$$

whose entries are in K, and a 1-1 and onto map h such that, the function

$$T: \alpha \mapsto W(\alpha) := H(\alpha)(zI - F(\alpha))^{-1} G(\alpha)$$

for $\alpha \varepsilon \Omega$, can be expressed as

$$T = hoP$$

on Ω, where "o" is the function composition. A function P as above is called a uniform parameterization of $\Sigma(\alpha)$.

2.2 DEFINITION. A parameterized system $\Sigma(\alpha)$ is canonically uniformly parameterized iff its transfer matrix

$$W(\alpha) := H(\alpha)(zI - F(\alpha))^{-1}(G(\alpha))$$

can be expressed as

$$W(\alpha) = N(\alpha)(d(\alpha) \cdot I)^{-1}$$

where $N(\alpha)$ is a polynomial matrix in z whose coefficients are functions of α, $d(\alpha)$ is a monic in z whose coefficients depend on α, and at each point $\alpha^* \varepsilon \Omega$, the entries of $N(\alpha^*)$ and $d(\alpha^*)$ have no common factor. (i.e., $d(\alpha^*)$ is the minimal re-

cursion polynomial for $W(\alpha^*)$).

2.3. REMARK. Thus $d(\alpha^*)$ is the least common denominator of $W(\alpha^*)$ at each $\alpha^* \varepsilon \Omega$. That is, $d(\alpha)$ is a uniform (global) least common denominator for $W(\alpha)$. The significance of such an $(N(\alpha),d(\alpha))$ stems from the following fact.
Express

$$N(\alpha) = \sum_{i=0}^{r-1} N_i(\alpha) z^i$$

and

$$d(\alpha) = \sum_{i=0}^{r-1} d_i(\alpha) z^i + z^r.$$

Then, the matrix

$$L(\alpha) := [d_o(\alpha) I : \ldots : d_{r-1}(\alpha) I : N_o(\alpha) : \ldots : N_{r-1}(\alpha)]$$

represents $W(\alpha)$, (in a canonical way) uniformly, in the sense that for each $\alpha^* \varepsilon \Omega$, there corresponds a unique set of values of the α-dependent entries of $L(\alpha^*)$, which is uniquely determined by $W(\alpha^*)$. Uniform parameterizations are needed for the existence of a single (adaptive) identifier (with a fixed structure) to be able to determine $W(\alpha^*)$ for any α^* in Ω.

We call a vector function $P(\alpha)$ whose entries are the distinct α-dependent entries of $L(\alpha)$ a canonical uniform parameterization of $\Sigma(\alpha)$.

2.4. LEMMA. If a canonical uniform parameterization for a system $\Sigma(\alpha)$ over K exists, then it is unique.

PROOF. At each $\alpha^* \varepsilon \Omega$ $W(\alpha^*)$ has a unique representation

$$W(\alpha^*) = \frac{N_{\alpha^*}}{d_{\alpha^*}} \tag{2.5}$$

where d_{α^*} is the least common denominator of the entries of $W(\alpha^*)$ and N_{α^*} is the unique polynomial matrix satisfying (2.5). Thus, at each $\alpha^* \varepsilon \Omega$ we have

$$N(\alpha^*) = N_{\alpha^*}$$

and

$$d(\alpha^*) = d_{\alpha^*}$$

for any pair $(N(\alpha),d(\alpha))$ satisfying (2.2). From the uniqueness of each d_{α^*} and N_{α^*}, the result follows. □

2.6. REMARK. It is easy to see from the results of [5], that the coefficients in a pair $(N(\alpha),d(\alpha))$ yield a canonical uniform parameterization of $W(\alpha)$ iff the $d(\alpha)$-realization of

$$d^{-1}(\alpha) h(\alpha)$$

is reachable for every $\alpha \varepsilon \Omega$. Here $h(\alpha)$ is obtained from $N(\alpha)$ by taking all nonzero entries of $N(\alpha)$ to form a row vector $\bar{h}(\alpha)$ and taking the remainder of $\bar{h}(\alpha)$ modulo $d(\alpha)$. These operations are straightforward to perform.

2.7. REMARK. For the identification of $\Sigma(\alpha)$ at a single point $\alpha^* \varepsilon \Omega$, it is enough to have a representation of $W(\alpha)$ as

$$W(\alpha) = \frac{N(\alpha)}{d(\alpha)}$$

where the entries of $N(\alpha^*)$ and $d(\alpha^*)$ have no common factor. (See [2]). However, to be able to use an identification technique in adaptive control/observers, we need to identify a parameter vector α^* which is allowed to change to any point in Ω, since the main purpose of our adaptive technique is to be able to obtain satisfactory performance for any variation in the parameter values within Ω. In fact, as the parameterization is continuous, if the above condition holds at a single point $\alpha^* \varepsilon \Omega$, it holds in an open neighborhood of α^*, U_1, in Ω. Then, redefining Ω to be U_1, we can always incorporate this situation into our framework.

The following is a common definition of identifiability.

2.8. DEFINITION. A parameterized system $\Sigma(\alpha)$ is identifiable on Ω iff the map

$$\alpha \mapsto W(\alpha)$$

is a 1-1 map on Ω.

The following result follows from Definitions 2.1, 2.8, and Lemma 2.4.

2.8a. COROLLARY. A uniformly parameterized system $\Sigma(\alpha)$ is identifiable on Ω iff the map

$$P:R^N \to R^M;\ \alpha \mapsto P(\alpha)$$

where $P(\alpha)$ is an M-vector which is any uniform parameterization of $\Sigma(\alpha)$, is a 1-1 map on Ω.

A key concept which is necessary for any globally convergent identification scheme to uniquely identify a linear system in the literature is that of a "sufficiently exciting" input u. (See [1-3]).

2.9. DEFINITION. A piecewise continuous input $u:[0,\infty) \to R^m$ is sufficiently exciting for $\Sigma(\alpha)$ on Ω iff for each α^* in Ω; for any linear system $\Sigma_1=(F_1,G_1,H_1)$, of the same order as $\Sigma(\alpha)$, $\Sigma(\alpha^*)$ and Σ_1 have the same zero-stated response to u implies

$$W(\alpha^*) = H_1(zI-F_1)^{-1}G_1.$$

That is, the transfer matrix of $\Sigma(\alpha^*)$ is uniquely determined from its zero-state response to u on $[0,\infty)$ for every α^* in Ω.

2.10. DEFINITION. A uniformly parameterized system $\Sigma(\alpha)$ is extendably identifiable iff it has a uniform parameterization $P(\alpha)$ which is an M-vector with $M \geq N$ and there exists a twice continuously differentiable 1-1 and onto function with a differentiable inverse (i.e., a twice continuously differentiable diffeomorphism)

$$P_1:R^M \to R^M;\ (\alpha,\beta) \mid \to P_1(\alpha,\beta)$$

such that the uniform parameterization P is a restriction of P_1 to $\Omega \times \mathbb{R}^{M-N}$, i.e., for each $\alpha^* \varepsilon \Omega$

$$P(\alpha^*) := P_1(\alpha^*,\beta^*)$$

for some $\beta^* \varepsilon \mathbb{R}^{M-N}$. A parameterized system is

canonically extendably identifiable iff it is canonically uniformly parameterized, and the above property holds for its canonical uniform parameterization $P(\alpha)$.

An example of a canonically extendably identifiable system is an implicitly parameterized system where the transfer matrix is of the form

$$W_1(z) = \frac{N_1(z)}{d_1(z)}$$

with all the entries of the coefficient matrices of $N_1(z)$ and all coefficients (except for the highest power) of the monic polynomial $d_1(z)$ are assumed to be free unknown parameters. It is assumed that at the unknown true values of these parameters, $d_1(z)$ is a least common denominator of the entries of $W_1(z)$ (see Anderson [2] for details). In this case

P_1 = identity map on R^M

P = identity map on Ω

Ω = some open set in R^N containing α^*, which is the complement of the zero set of a finite number of polynomials.

In general, if a parameterized system $\Sigma(\alpha)$ is canonically extendably identifiable then there exists a transfer matrix

$$W(\alpha,\beta) = \frac{N(\alpha,\beta)}{d(\alpha,\beta)}$$

such that for each α^* in Ω,

$$H(\alpha^*)(zI-F(\alpha^*))^{-1}G(\alpha^*) = \frac{N(\alpha^*,\beta^*)}{d(\alpha^*,\beta^*)}$$

for some $\beta^* \in R^{M-N}$.

2.11. REMARK. It is a consequence of the rank theorem that, any continuously differentiable function which is locally 1-1 in an open set U, with a Jacobian matrix whose columns are linearly independent on U, is the restriction of a diffeomorphism in a neighborhood of points of U. Thus, a locally identifiable uniformly parameterized system is locally extendably identifiable in a neighborhood of each point in its parameter space.

Next, it is shown that, if $\Sigma(\alpha)$ is a canonically extendably identifiable system, then there exist asymptotic parameter identification techniques to estimate α^*, under assumptions which are the same as those used for single input-single output implicitly parameterized systems (see e.g., [1]).

Let $\Sigma(\alpha)$ be a canonically extendably identifiable system with its transfer matrix

$$W(\alpha) = H(\alpha)(zI-F(\alpha))^{-1}G(\alpha)$$

$$= \frac{N(\alpha)}{d(\alpha)} ,$$

where $(N(\alpha),d(\alpha))$ forms the canonical extendably identifiable

parameterization of $W(\alpha)$. A wellknown realization of $W(\alpha)$ as a linear system over K is given by $\Sigma_1(\alpha) := (F_1(\alpha), G_1(\alpha), H_o)$,

$$\dot{x}_{o\alpha} = F_1(\alpha)x_{o\alpha} + G_1(\alpha)u$$

(2.12)

$$; x_o(0) = x_\alpha,$$

$$y_\alpha = H_o x_{o\alpha}$$

where

$$F_1(\alpha) := \begin{bmatrix} 0 & 0 \ldots . 0 & -d_o \cdot I \\ I & 0 \quad . & -d_1 \cdot I \\ 0 & I \quad . & . \\ . & 0 \quad . & . \\ . & . \quad . & . \\ . & . \quad . & . \\ . & . \quad . & . \\ 0 & 0 \ldots 0 \; I & -d_{r-1} \cdot I \end{bmatrix}$$

$$G_{o1}(\alpha) := \begin{bmatrix} N_o \\ . \\ . \\ . \\ . \\ N_{r-1} \end{bmatrix}$$

$$H_o = [0 \ldots 0 I].$$

Such a realization is also used in [2] for identification of implicitly parameterized systems, where all the entries of N_j's and d_i's are considered to be independent variables on $\mathbb{R}^N$.

Let

$$k(z) := z^r + k_{r-1}z^{r-1} + \ldots + k_o$$

be any polynomial with roots in the open left half complex plane. Clearly, the zero state responses of $\Sigma(\alpha)$ and $\Sigma_1(\alpha)$ are the same at every $\alpha \in \Omega$. We can rewrite (2.12) as

$$\dot{x}_{o\alpha} = F_o x_{o\alpha} + G_2(\alpha)y + G_1(\alpha)u$$

(2.13)

$$y_\alpha(t) = H_o x_{o\alpha}; \; x_{o\alpha}(0) = x_\alpha$$

where F_o is the same form as $F_1(\alpha)$ except that d_i is replaced

by k_i, and

$$G_{o2}(\alpha) := \begin{bmatrix} (k_o - d_o)\cdot I \\ \vdots \\ (k_{r-1} - d_{r-1})\cdot I \end{bmatrix}$$

Let

$$y_\alpha = \begin{bmatrix} y_{1\alpha} \\ \vdots \\ y_{p\alpha} \end{bmatrix}, \quad u = \begin{bmatrix} u_{1\alpha} \\ \vdots \\ u_{m\alpha} \end{bmatrix}$$

Then (3.13) can be rewritten as

$$\dot{x}_{o\alpha} = F_o x_{o\alpha} + [y_{1\alpha} I : \ldots : y_{p\alpha} I] \begin{bmatrix} g_1 \\ \vdots \\ g_p \end{bmatrix} + [u_{1\alpha} I : \ldots : u_{m\alpha} I] \begin{bmatrix} h_1 \\ \vdots \\ h_m \end{bmatrix}, \tag{2.14}$$

$$y_{o\alpha} = H_o x_{o\alpha},$$

where g_i and h_i denote the i-th columns of $G_2(\alpha)$ and $G_1(\alpha)$ respectively.

Defining

$$V_{1\alpha}(t) := H_o \int_o^t e^{F_o(t-\zeta)} [y_{1\alpha}(\zeta) I : \ldots : y_{p\alpha}(\zeta) I : \ldots : u_{m\alpha}(\zeta) I] d\zeta, \tag{2.15}$$

we obtain

$$y_\alpha(t) = V_{1\alpha}(t) \begin{bmatrix} g_1(\alpha) \\ \vdots \\ g_p(\alpha) \\ h_1(\alpha) \\ \vdots \\ h_m(\alpha) \end{bmatrix} + H_p e^{F_o t} x_\alpha \tag{2.16}$$

$$= V_{1\alpha}(t)\gamma(\alpha) + H_o e^{F_o t} x_\alpha$$

In (2.16), in general, some of the entries in $\gamma(\alpha)$ will be identical or zero. Taking this into consideration, we can rewrite (2.16) as

$$(2.17)\qquad y_\alpha(t) = V_{l\alpha}(t)(\bar{P}(\alpha) + k) + H_o e^{F_o t} x_\alpha$$

where the entries of k are determined (and known) by the coefficients of the chosen polynomial $K(z)$, and the entries of $\bar{P}(\alpha)$ are the unknown entries of $\gamma(\alpha)$. Thus, (2.17) can be rewritten as

$$y_\alpha(t) = V_{l\alpha}(t)\bar{P}(\alpha) + y_{l\alpha}(t) + H_o e^{F_o t} x_\alpha,$$

where $y_{l\alpha}$ is a known function determined by $V_{l\alpha}$ and k. The entries of the vector $\bar{P}(\alpha)$ are the entries of $L(\alpha)$. Some of these entries can be known and some of them can be identical. Thus, we can rewrite (2.18) as

$$(2.18)\qquad \tilde{y}_\alpha(t) = V_\alpha(t)P(\alpha) + H_o e^{F_o t} x_\alpha$$

where $\tilde{y}_\alpha$, V_α are known time functions, and the entries of $P(\alpha)$ are the distinct α-dependent entries of $L(\alpha)$. From (2.17), (2.18) and the fact that $\Sigma(\alpha)$ is uniformly identifiable on Ω, it follows that the input u is sufficiently exciting iff the columns of $V_\alpha(t)$ (as functions of time on $[0,\infty)$) are linearly independent over $\mathbb{R}$ for any $\alpha\in\Omega$. Also, it follows that the map $\alpha \to \tilde{y}_\alpha$ is a 1-1 map iff P is 1-1 and u is sufficiently exciting. Since $\Sigma(\alpha)$ is extendably identifiable, there exists an integer M, $M \geq N$, and there exists a diffeomorphism

$$P_1 : R^M \to R^M;\ (\alpha,\beta) \to P_1(\alpha,\beta)$$

which is twice continuously differentiable, such that for each $\alpha^*\in\Omega$

$$P(\alpha^*) = P_1(\alpha^*,\beta^*)$$

for some β^* in $\mathbb{R}^{M-N}$. Clearly V_α can be easily generated from given u and y as output of a known linear system.

If we assume $\Sigma(\alpha)$ to be implicitly parameterized system, in the sense that we are only concerned with determining $P(\alpha^*) = P_1(\alpha^*,\beta^*)$ and/or x_{α^*}, but not necessarily α^* itself, (2.18) is in a common form used for the system identification techniques appearing in the literature (see, e.g., [1-2]).

For the following first parameter estimation technique, we will assume $\Sigma(\alpha^*)$ is at zero state, and we will choose x_{α^*} to be identically zero. However, this technique can be easily modified for the case where $\Sigma(\alpha^*)$ may not be at zero initial state (see Remark 2.21). The following theorem is an extension to the nonlinear case of the techniques used in [1] for implicitly parameterized single input-single output systems. It is an extension to a parameter estimation technique for multivariable, canonically extendably identifiable continuously parameterized systems, and can be viewed as a time-varying Newton-like techniuqe. Our extension here unavoidably leads to nonlinear differential equations. As such, the existence of solutions on $[0,\infty)$ as well as convergence are considerably more involved than the linear case established in [1].

2.19. THEOREM. Let $\Sigma(\alpha)$ be a completely uniformly parameterized, extendably identifiable system. Let u be a sufficiently rich piecewise continuous input applied to $\Sigma(\alpha^*)$, where $\alpha^* \in \Omega$ is the true value of the parameter vector. Suppose $\Sigma(\alpha^*)$ is at zero initial state. Then there exists a continuous function $\alpha(t):[0,\infty) \to R^N$, such that

$$\alpha(t) \to \alpha^* \text{ as } t \to \infty.$$

Furthermore, such a function can be generated by solving a differential equation initialized arbitrarily.

PROOF. Consider the differential equations

$$\begin{bmatrix} \dot{\alpha}(t) \\ \dot{\beta}(t) \end{bmatrix} = -L(\alpha(t),\beta(t))R(t)P_1(\alpha(t),\beta(t)) + L(\alpha(t),\beta(t))r(t), \tag{2.20}$$

$$\begin{bmatrix} \alpha(0) \\ \beta(0) \end{bmatrix} = \text{arbitrarily chosen in } R^M,$$

$$\dot{R}(t) = V_{\alpha^*}^T(t)V_{\alpha^*}(t);\ R(0) = 0,$$

$$\dot{r}(t) = V_{\alpha^*}^T(t)\tilde{y}_{\alpha^*}(t);\ r(0) = 0,$$

$$L_1(\alpha(t),\beta(t)) := \left[\frac{\partial P_1(\alpha(t),\beta(t))}{\partial \alpha} : \frac{\partial P_1(\alpha(t),\beta(t))}{\partial \beta} \right],$$

$$L(\alpha(t),\beta(t)) := L_1(\alpha(t),\beta(t))^{-1}$$

It is shown in [4] that for each $(\alpha(0),\beta(0))$, (2.20) has a unique solution on $[0,\infty)$ which is bounded, and that $\alpha(t) \to \alpha^*$ and $\beta(t) \to \beta^*$ as $t \to \infty$. □

2.21. REMARK. The identification technique in Theorem 2.19 can be extended to the case where the initial state x_{α^*} is not zero as follows. Suppose $\Sigma(\alpha)$ is canonically extendably identifiable, $\Sigma(\alpha^*)$ is possibly unstable, and possibly at a nonzero initial state, and reachable at every point of the parameter space. Then, to every initial state of $\Sigma(\alpha^*)$, there corresponds an initial state $x_{0\alpha^*}$ of $\Sigma_1(\alpha^*)$ so that both produce the same outputs under the same inputs. Then, in Theorem 2.19, we can replace $V_{\alpha^*}(t)$ by $[V_{\alpha^*}(t):H_0e^{F_0t}]$, as done in [1]. Then throughout Theorem 2.19, replace $[\alpha(t):\beta(t)]$ by $[\alpha(t),\beta(t),c(t)]$, and L_1 by $\begin{bmatrix} L_1 & 0 \\ 0 & I \end{bmatrix}$. The same reasoning in Theorem 2.19 shows that the resulting differential equation in $[\alpha(t),\beta(t),c(t)]$ will converge to $[\alpha^*,\beta^*,x_{0\alpha^*}]$ provided u is such that the columns of $[V_{\alpha^*}(t):H_0e^{F_0t}]$ are linearly independent over $\mathbb{R}$. This idea has been used in Kreisselmeier [1] for single input-single output implicitly parameterized systems for linear identification.

2.22. REMARK. The results presented here can be used for parameter adjustment in some globally stable adaptive control schemes, and adaptive observers. For this we refer to [4].

Some simple examples of parameterized systems for which the results of [4] yield globally stable Adaptive Stabilization/pole-placement or Adaptive Linear Quadratic Optimal control techniques are

i) implicitly parameterized systems $\Sigma(\alpha) = (F(\alpha),G(\alpha),I)$ where α is the vector of the unknown entries of $F(\alpha)$ and $G(\alpha)$; which are reachable for each α in $\mathbb{R}^N$.

ii) single input-output systems whose transfer functions are given as

$$W = \frac{b_{n-1}s^{n-1} + \ldots + b_o}{s^n + a_{n-1}s^{n-1} + \ldots + a_o} = \frac{b(s)}{a(s)}$$

where the unknown ones among a_i's and b_i's are assumed to be independent unknowns, and for every real values of the unknowns, any common factor of $b(s)$ and $a(s)$ has its roots in the left half-plane.

3. REFERENCES

[1] G. Kreisselmeier, "Algebraic separation realizing a linear state feedback control law by means of an adaptive observer," IEEE Trans. Autom. Contr., Vol. AC-25, No. 2, pp. 238-243, April 1980.

[2] B.D.O. Anderson, "An approach to multivariable system identification," Automatica, Vol. 13, pp. 401-408, 1977.

[3] J.S-C. Yuan and W.M. Wonham, "Probing signals for model reference identification," IEEE Trans. Autom. Contr., Vol. Ac-22, pp. 530-538, August 1977.

[4] E. Emre, "Adaptive control/observers and asymptotic parameter identification of multivariable systems," Sept. 1984, submitted for publication.

[5] E. Emre and P.P. Khargonekar, "Regulation of split linear systems over rings: coefficient assignment and observers," IEEE Trans. Autom. Contr., Feb. 1982.

ON MODEL REFERENCE ADAPTIVE CONTROL (MRAC) AND IDENTIFICATION - ALGORITHMS TO EASE IMPLEMENTATION

PETER J. KENNEDY
AT&T Information Systems
Lincroft, NJ

ABSTRACT

An MRAC method is presented for linear, deterministic, time-invariant and minimum phase systems which adjusts estimated plant parameters; providing control and identification data. A priori plant information can be easily utilized to minimize hardware design. The system is uniformily, asymptotically stable using a parameter update algorithm that requires both the output and equation errors. A modification of this algorithm, for which the system is stable, is also presented which requires only the sign of the filtered I/O variables and the respective errors.

INTRODUCTION

In [1] an MRAC method was described utilizing a control law with the plant parameters readily accessible. The parameter update algorithm required both the output and equation errors. SECTION 1 of this short paper briefly summarizes [1]. SECTION 2 discusses a modification to the parameter update algorithm presented in [1].

SECTION 1: SYSTEM DESIGN

In [1] the plant output $y(t)$, and the plant input $u(t)$ are related by:

$$y(t) = G(p)u(t) = [N(p)/M(p)]u(t) \tag{1}$$

where p is the differential operator, M and N are polynomials with elements a_i and b_i respectively, and orders m and n with $m \geq n$ and $a_m = 1$. The model output $y_m(t)$ is related to the bounded plant input $r(t)$ by:

$$y_m(t) = T(p)r(t) = [N_T(p)/M_T(p)]r(t) \tag{2}$$

where N_T and M_T are strictly Hurwitz polynomials of orders n_T and m_T respectively.

With the plant parameters unknown, the plant output follows the model output as $t \to \infty$ using the control algorithm:

$$u(t) = -\phi^{-1}[\hat{N}_1 u_f(t) - \hat{M} y_f(t)] + K_0[r_1(t) - T^{-1}y_1(t)] \tag{3}$$

where $y_f = D_1^{-1}y$, $u_f = D_1^{-1}u$, $y_1 = D^{-1}y$, $r_1 = D^{-1}r$; ϕ, D, and D_1 are Hurwitz polynomials of orders 1, m-n, and m; K_0 is a positive gain constant; and $\hat{N}_1 = \hat{N} - \phi D_1$. The tilde represents polynomials with the estimated plant parameters. It is assumed $(m_T - n_T) \geq (m-n)$.

It is shown in [1] that the output error is related to the equation error by a strictly positive real transfer function, or:

$$e_0(t) = K_0^{-1}[(DN_T)/(\emptyset M_T)]e_1(t) = y(t) - y_m(t) \quad (4)$$

and $$e_1(t) = Z(t)^T X(t) \text{ - equation error} \quad (5)$$

where $x_i(t) = \{p^i y_f(t), p^i u_f(t)\}$; $z_i(t) = \{\delta a_i(t), \delta b_i(t)\}$ - plant parameter errors.

In [1] the uniform, asymptotic stability of the system is proven by defining a Lyapanov function

$$V = 1/2\ E^T PE + \frac{1}{2} Z^T \wedge Z \quad (6)$$

E is an m+1 vector obtained by putting EQ. 6 in SVF, X(t) and Z(t) are n+m+1 vectors and $\wedge$ is a positive definite diagonal matrix. Using the MKY lemma one obtains

$$\dot{V} = -\frac{1}{2} E^T QE + e_0 e_1 + Z^T \wedge \dot{Z} \quad (7)$$

and finally the proof is completed using the parameter update algorithm

$$\dot{Z} = -\wedge^{-1} X(t)[e_0(t) + K_1 e_1(t)] \quad (8)$$

where K_1 is a positive gain constant.

SECTION 2

Modified Parameter Update Algorithm - EQ. 8 is modified using some of the ideas expressed in [2] as follows:

$$\dot{Z} = -\wedge^{-1} R(t)[K_3(t) e_0 \operatorname{sgn}(w) + e_1] \quad (9)$$

where $R_i(t) = \operatorname{sgn}(x_i)$; $w = e_0 e_1$; $K_3(t) \geq \max|x_i(t)|$ at any t.

Stability is proven by defining $R = SX$ where S is a diagonal matrix with $\{|x_i|^{-1}\}$ and substituting EQ.'s 5 and 9 into EQ. 7 one obtains

$$\dot{V} = -[\tfrac{1}{2} E^T QE + Z^T SXX^T Z] + e_1 e_0 - K_3(t) Z^T R e_1 |e_1|^{-1} (\operatorname{sgn} e_0) e_0 \quad (10)$$

Expanding the last two terms of EQ. 10 one obtains

$$\dot{V} = -[\tfrac{1}{2} E^T QE + Z^T SXX^T Z + Z^T GXX^T Z]$$

where $-G = [e_0 e_1^{-1} I - |e_0||e_1|^{-1} K_3(t) S]$ which is a negative semidefinite matrix since $K_3(t) \geq \max|x_i(t)|$. Since XX^T is nonnegative definite we have $\dot{V} \leq 0$ which establishes the boundedness of e_0 and Z. This algorithm should minimize the hardware implementation and cost since it eliminates multiplications between the filtered I/O variables and the error terms.

REFERENCES

1. P. J. Kennedy, "On Model Reference Adaptive Control and Identification," 1984 AACC Proceedings, pgs. 1252-1258, June 1984.

2. A. A. Abdul Karim, R. Nagarajan, "Continuous Time Relay-Controlled Model Reference Adaptive System," Proceedings IEEE, Vol. 21, No. 2, February 1983.

CONTINUOUS STABILIZATION OF NONLINEAR AND FAMILIES OF LINEAR TIME-VARYING SYSTEMS†

by

E. Emre and G. Knowles
Dept. of Electrical Engineering/Computer Science
Texas Tech University
Lubbock, TX 79409

0. ABSTRACT

Stabilization of families of linear time-varying systems and a class of nonlinear time-varying systems is considered. It is shown that, if a family of linear time-varying systems is parameterized continuously by a parameter set T (a topological space of a certain class including all metric spaces), then there exists a family of stabilizing linear time-varying state-feedback laws continuously parameterized by T provided the family is uniformly completely controllable at each point of T, and σ-continuous. If, in addition, the family is equi-uniformly completely controllable on T and satisfies some boundedness conditions, then the steady state solution of the control Riccati equation is shown to be a continuous function on T. These results and their natural dualizations show the existence of continuously parameterized families of optimal controllers and Kalman filters for a class of continuous families of linear time-varying systems. These results together with fixed point theory are shown to lead to new global as well as local state feedback stabilization criteria (with arbitrary rate of decay) for a class of nonlinear time-varying functional differential equations.

1. INTRODUCTION

In this paper we summarize some of the results of [18] on two main problems of control theoretic interest which, in our setting, are very closely related. The first one is the problem of stabilization of families of linear time-varying systems parameterized continuously by a metric space T, by using a family of controllers again parameterized by T continuously. The second problem is the problem of determining local and global stabilization criteria for nonlinear time-varying systems of the type

$$(1.1) \qquad \dot{x}(t)=F_x(t)x(t) + G_x(t)u(t),$$

where F,G are continuous functions of the state, x, by state-feedback. This paper gives conditions for local and global stabilization of (1.1) in terms of the family of linear time-varying systems.

$$(1.2) \qquad \dot{x}_\omega(t)=F_\omega(t)x_\omega(t) + G_\omega(t)u_\omega(t)$$

continuously parameterized by ω which is in certain subsets of the set of continuous functions from $[0,\infty)$ to $\mathbb{R}^n$, denoted as

†This research was supported in part by the U.S. Air Force Office of Scientific Research Grant AFOSR-82-0282, in part by the ONR Research Contract N00014-84-C-0104, and in part by the NSF Research Grant ECS-8204512.

C. F and G are as in (1.1), and thus depend on ω continuously. As F,G can be arbitrary continuous functions of ω, our results are valid for a class of functional differential equations which includes a class of nonlinear time-varying systems with delays. In order to obtain our results on stabilization of (1.1), in [18] we first (necessarily) develop several results on stabilization of (1.2) by a family of linear time-varying state feedback controllers

$$(1.3) \qquad u^c_\omega(t) = -L_\omega(t)x^c_\omega(t),$$

where L is a continuous function of ω. Then we combine our results with Tychonoff's fixed point theorem (see Warga [14]) to obtain local and global stabilization criterial for (1.1), thus being able to exploit results developed for linear systems to obtain new results for nonlinear systems. It is shown in [18] that equi-uniform complete controllability and uniform boundedness of F,G (see Section 2) of (1.2) on some subsets of C is sufficient for exponential stabilizability with exponential rate of decay of the system (1.1) by continuous state feedback. Our results are also applicable to some nonlinear time-varying systems described by functional differential equations containing delays (see [18]). We also show in [18] that a Newton-type algorithm of Kleinman [9], developed for computing the steady state solution of the Algebraic Riccati Equation for time-invariant systems, can also be extended to linear time-varying case. This provides a technique for fast approximation of the steady state solution of the Riccati equation in the time-varying case, in terms of linear Lyapunov equations, as well as a theoretical tool to study Riccati equations for time-varying systems. It follows from our results that there exist families of optimal controllers and Kalman filters which are also continuous functions of ω(for continuously parameterized uniformly bounded families which are equi-uniformly completely controllable and observable at each point ω). These results provide an extension of some recent results in the time-invariant case in a systems over rings setting (see, e.g., Emre [2], Delchamps [3], Rodman [17], and the references there) for parameterized systems which are pointwise reachable and observable at points of $\mathbb{R}^n$ or $\mathbb{C}^n$. In the time-invariant case the families of systems has been thus far considered mainly in the cases where the parameter set (where ω belongs to) is usually $\mathbb{R}^n$ or $\mathbb{C}^n$. In our setting ω can be an element of any metric space (in fact any topological space where sequential continuity is equivalent to continuity). Also, here we consider families of time-varying systems. One of the reasons for having to allow such a general parameter set is that for our nonlinear stabilization results ω will be an element of C. In our setting, C is a complete metric space as well as a locally convex linear topological space, but it is not a normed space. Since we are concerned with time-varying systems, our techniques are considerably different from the previous techniques utilized for the time-invariant case. Previously, Schauder's fixed point theorem has been used in Davison and Kunze [15], to obtain (finite interval) controllability criteria for a class of systems as in (1.1). However, the stabilization problem (or our families of systems results) have not been considered in [15]. Here,

since we consider functions on $[0,\infty)$, we are necessarily led to the consideration of families of linear systems first, and we necessarily use Tychonoff's fixed point theorem instead of the Schauder's (the first being a generalization to locally convex spaces of the latter). In the somewhat more general case where F and G in (1.1) are functions of not only x but also of u, we obtain some similar (open-loop) control results as well (see [18]). Some results of [15] (based on certain results of [15,16]) can be easily modified to obtain some classes of nonlinear systems which satisfy our sufficient conditions. A more general approach to equi-uniform complete controllability follows from Silverman and Meadows [16], and Davison and Kunze [15].

2. STABILIZATION OF CONTINUOUS FAMILIES OF LINEAR TIME-VARYING SYSTEMS

In this section we consider families of linear time-varying systems of the form

$$\dot{x}_\omega(t)=F_\omega(t)x_\omega(t) + G_\omega(t)u_\omega(t) \tag{2.1}$$

$$y_\omega(t)=H_\omega(t)x_\omega(t) \tag{2.2}$$

where F,G,H are, respectively functions of ω given as

$$F:T \to C((-\varepsilon,\infty),\mathbb{R}^{n\times n});\ \omega \mapsto F_\omega$$

$$G:T \to C((-\varepsilon,\infty),\mathbb{R}^{n\times m});\ \omega \mapsto G_\omega$$

$$H:T \to C((-\varepsilon,\infty),\mathbb{R}^{p\times n});\ \omega \mapsto H_\omega.$$

Here, T is any metric space, $\mathbb{R}_+$ is nonnegative real numbers C(A,B) denotes continuous functions from the metric space A to the metric space B. We will consider C(A,B) as a linear topological space always with the compact convergence (compact open) topology (i.e., a sequence of functions in C converges to a function f iff it converges uniformly on each compact subset of A) (see, e.g., [1] for details of this topology). We will assume that F,G,H are continuous maps with respect to the topology specified above. Thus, for each given ω in T, (2.1) and (2.2) represents a known linear time-varying system whose parameters are continuous functions defined on $(-\varepsilon,\infty)$. Here, $\varepsilon > 0$ is an arbitrary fixed small number which we introduce for technical reasons so that for each ω the corresponding system is well defined on an open subset Ω of $\mathbb{R}^{n+1}$ which includes $\mathbb{R}^n\times\mathbb{R}_+$. Since each continuous real valued function f on $[0,\infty)$ can be extended to a continuous function on $(-\varepsilon,\infty)$ by connecting $-\varepsilon$ and f(0) by a straight line, the compact convergence properties of $C([0,\infty),\mathbb{R}^n)$ preserved, and thus our results are applicable to the case where functions are defined only for $t \geq 0$. Clearly, this is a minor technicality.

Also, we could have picked any t_0 in $\mathbb{R}$ instead of 0 as initial time, and all of our results are equally applicable to that case. Thus (2.1) and (2.2) is a family of time-varying systems indexed (parameterized) by the metric space T. We will denote such a family by Σ, or $\Sigma_\omega=(F_\omega,G_\omega,H_\omega)$.

The concepts of uniform complete controllability and uniform complete observability have originated in Kalman [4].

For a fixed ω in T, the system Σ_ω, or the pair (F_ω, G_ω) is a uniformly completely controllable (UCC) [4] iff there exists a real number $\sigma(\omega) > 0$ such that

$$\text{(i)} \qquad 0 < \alpha_0 I < W_\omega(t,t+\sigma) \leq \alpha_1 I;\ t \geq 0,$$

$$\text{(ii)} \qquad 0 < \beta_0 I \leq \Phi_\omega(t+\sigma,t)W(t,t+\sigma)\Phi_\omega^T(t+\sigma,t) \leq \beta_1 I;\ t \geq 0,$$

for some real numbers $a_0(\omega,\sigma(\omega)), a_1(\omega,\sigma(\omega)), \beta_0(\omega,\sigma(\omega))$, $\beta_1(\omega,\sigma(\omega))$. Here $W_\omega(t,t+\sigma)$ is defined as [4]

$$W_\omega(t,t+\sigma) := \int_t^{t+\sigma} \Phi_\omega(t,\zeta)G_\omega(\zeta)G_\omega^T(\zeta)\Phi_\omega^T(t,\zeta)\,d\zeta$$

where $\Phi_\omega(t,\zeta)$ is the state transition matrix for Σ_ω. We note that (i) and (ii) automatically give

$$\text{(iii)} \qquad \|\Phi_\omega(t,\zeta)\| \leq \alpha_\omega(|t-\zeta|),$$

where $\alpha_\omega : \mathbb{R}_+ \to \mathbb{R}_+$ is bounded on bounded sets. Indeed any two of (i), (ii), (ii) imply the third [4]. Clearly, for each ω, if (F_ω, G_ω) is UCC, there are many choices of $\sigma(\omega)$ and a_i's, b_i's as above. A case of importance to us is when $\sigma(\omega)$, $\omega \in T$, can be chosen such that the function $\sigma : T \to \mathbb{R}_+$ is a continuous function. We call the family $(F_\omega, G_\omega); \omega \in T$, σ-continuous if such a choice is possible. Clearly, this would be the case if a σ could be chosen which is independent of ω on T.

It follows from the results of Kwon and Pearson [8] that for a fixed ω, the feedback law

$$(2.4) \qquad L_\omega(t) := G_\omega^T(t)W_\omega^{-1}(t,t+\sigma(\omega));\ t \geq 0,$$

is a stabilizing feedback law for the pair (F_ω, G_ω), provided it is uniformly completely controllable. It can be proved that

2.5. THEOREM. If (F_ω, G_ω), a continuously parameterized family which is uniformly completely controllable for each $\omega \in T$, and if it is σ-continuous, then the feedback law (2.4) is a stabilizing feedback law which depends on ω continuously as a map $L : T \to C([0,\infty), \mathbb{R}^{m\times n})$; $\omega \mapsto G_\omega^T(\cdot)W^{-1}(\cdot,\cdot+\sigma(\omega))$, with the compact convergence topology in its codomain.

2.6. REMARK. In [8], there is a more general class of stabilizing feedback laws given which can be shown to depend on ω continuously similar to the proof of Lemma 2.5. However, in this paper we start with (2.4) for its simplicity, and for simplicity of exposition of our results.

Next, we will consider the continuity with respect to of the parameterized Riccati equation

$$(2.7) \qquad -\frac{d}{dt}P_\omega(t) = F_\omega^T(t)P_\omega(t) + P_\omega(t)F_\omega(t) - P_\omega(t)G_\omega(t)G_\omega^T(t)P_\omega(t) + I$$

It is shown in Kalman [4], that, if for a fixed $\omega \in T$, the system Σ_ω is uniformly completely controllable, then the solution $\Pi_\omega(t,0,t_1)$ of (2.8) subject to

$$\Pi_\omega(t_1,0,t_1) = 0$$

exists in $[0,t_1+\varepsilon_1)$ for some $\varepsilon_1 > 0$. Also, it is shown in [4] that, for a fixed $\omega \in T$,

$$\bar{P}_\omega(t) := \lim_{t_1 \to \infty} \Pi_\omega(t,0,t_1)$$

exists for all $t \in [0,\infty)$, and it is a solution of (2.8). Furthermore, the closed-loop system (2.3) with the feedback law

$$L_\omega(t) := G_\omega^T(t)\bar{P}_\omega(t), \quad t \geq 0,$$

is exponentially stable for $t \geq 0$. Thus, there is a well-defined function

$$\bar{P}: T \to C([0,\infty), \mathbb{R}^{n\times n}); \ \omega \mapsto \bar{P}_\omega.$$

In [18], we prove that if Σ is equi-uniformly completely controllable and uniformly bounded, then $\bar{P}_\omega(t)$ is a continuous function of ω. This is done in [18] by generalizing a Newton-like algorithm of Kleinman [9], developed for time-invariant systems, to linear time-varying systems.

2.8. DEFINITION. A family of pairs (F_ω, G_ω) is equi-uniformly completely controllable with respect to a set T iff there exist positive constants, $a_o, a_1, b_o, b_2, \sigma$ such that

i) $$0 < a_o I \leq W_\omega(t,t+\sigma) \leq a_1 I$$

ii) $$0 < b_o I \leq \Phi_\omega(t,t+\sigma) W_\omega(t,t+\sigma) \Phi_\omega^T(t,t+\sigma) \leq b_1 I,$$

for all ω in T.

Thus, this is the same as the definition of uniform complete controllability in Kalman [4], except that we require the constants to be independent of ω in T.

2.9. THEOREM. If the continuous family (F_ω, G_ω) is equi-uniformly completely controllable and uniformly bounded on T, then the function

$$\bar{P}: T \to C([0,\infty), \mathbb{R}^{n\times n});$$
$$\omega \mapsto \bar{P}_\omega,$$

is a continuous function, with compact convergence topology on its codomain.

2.10. COROLLARY. If there exists an open covering UU_α of of T such that (F_ω, G_ω) is equi-uniformly completely controllable and uniformly bdd on each U_α, then $\bar{P}$ is continuous on T.

2.11. COROLLARY. With the above assumptions, for each fixed t in $[0,\infty)$, $V_{k\omega}(t)$ converges to $\bar{P}_\omega(t)$, all considered as functions of ω, uniformly on compact subsets of T.

2.12. REMARK. It follows from our results and the results of Ikeda et. al [6] that, application of our results to

$$\dot{x}_\omega(t) = (F_\omega(t) + m \cdot I) x_\omega(t) + G_\omega(t) u_\omega(t),$$

where $m > 0$, yields a continuously parameterized family of feedback laws of the type obtained in Theorems 2.5 and 2.9 . These can be used to obtain a family of closed-loop systems

whose response decays by any desired exponential rate regardless of ω in T for such families of linear time-varying systems. Thus, one can obtain continuously parameterized feedback laws which serve similar purpose as in pole-placement for families of linear time-invariant systems.

2.12. REMARK. The uniform boundedness (by a constant) condition on the family (F_ω, G_ω) in Theorem 2.9 can be replaced by weaker conditions such as uniform boundedness of only $G_\omega(t)$ by a continuous function $f(t) \geq 0$, $t \geq 0$, (such as a polynomial function) such that for any positive number m,

$$\int_0^\infty f(t)e^{-mt}dt$$

exists. In this case, again, the same result can be obtained. (See [18]).

3. STABILIZATION OF NONLINEAR TIME-VARYING SYSTEMS

In this section we consider nonlinear systems described by nonlinear functional differential equations (see Warga [14]) of the type

$$\dot{x}(t)=F_x(t)x(t) + G_x(t)u(t) \tag{3.1}$$

$$x(0)=x_0=\text{given}.$$

In (3.1), F and G are continuous functions of x from $C([0,\infty), \mathbb{R}^n)$ to $C((-\varepsilon,\infty), \mathbb{R}^{n\times n})$, and to $C((-\varepsilon,\infty), \mathbb{R}^{n\times m})$, respectively, where all three spaces are considered with the compact convergence topology. We should note that the (nonlinear) functions F,G in (3.1) are substantially general types of functions for nonlinear systems and include the more usual type where

$$F:\mathbb{R}^n\times\mathbb{R}_+ \to \mathbb{R}^{n\times n}$$

(3.1a)

$$G:\mathbb{R}^n\times\mathbb{R}_+ \to \mathbb{R}^{n\times m}$$

are continuous functions, and (3.1) is obtained as

$$\dot{x}(t)=F(x(t),t)x(t) + G(x(t),t)u(t) \tag{3.1b}$$

with

$$F_x(t)=F(x(t),t), \text{ and } G_x(t)=G(x(t),t).$$

For systems as in (3.1), we develop new local as well as global stabilization criteria. These results are based on our results on families of linear time-varying systems and the Tychonoff's fixed point theorem for locally convex linear topological spaces.

In what follows Ω is the set of all $\mathbb{R}^n$-valued Lipschitz continuous functions on $[0,\infty)$, f, such that for some positive constants d and $\tilde{m}$

$$\|f(t)\| \leq de^{-\tilde{m}t};\ t \geq 0.$$

3.2. THEOREM. Suppose that F,G in (2.1) are uniformly bounded on Ω. Let the family of systems (2.1) be equi-uni-

formly completely controllable with respect to Ω. Then there exists a continuous feedback law $L:\Omega \to C(\mathbb{R}_+, \mathbb{R}^{m\times n})$; $\omega \to L_\omega$ such that

$$u_c(t) = -L_{x^c(t)} x^c(t); \quad t \geq 0,$$

stabilizes the system (3.1) for any x_o in $\mathbb{R}^n$, with $x^c \in \Omega$. The fixed point theory and the results in Section 2 enable us to obtain local stabilization results as well.

3.12. THEOREM. Let $K > 0$ be a real number. Let $\Omega_K = \{f \in \Omega \mid \|f(t)\| \leq K \text{ for } t \geq 0\}$. Then there is a real number $K_1 > 0$ such that whenever $x_o < K_1$, then there exists a continuous feedback law $L:\Omega_K \to C(\mathbb{R}_+, \mathbb{R}^{m\times n})$; $\omega \to L_\omega$ which stabilizes the the system (3.1), as in (3.2). It is shown in [18] that our results also yield continuous feedback type local and global stabilization results (similar to Theorems 3.2 and 3.3) with arbitrary rate of exponential decay. Furthermore, our approach and results are also applicable to certain nonlinear systems described by nonlinear functional differential equations with delays. For these and some other results we refer to [18].

4. REFERENCES.

[1] J.R. Munkres, Topology, A First Course, Prentice-Hall, 1975.

[2] E. Emre, "On necessary and sufficient conditions for stabilizability of linear systems over rings," SIAM J. Contr., and Opt., March, 1982.

[3] D.F. Delchamps, "Analytic stabilization and the Riccati Equation," Proc. IEEE Conf. on Decis. and Contr., pp. 1396-1401, 1983. (See also D.F. Delchamps, "A note on the analyticity of the Riccati matrix," Algebr. and Geom. Meth.s in Linear Systems Theory, AMS, Vol. 18, 1980, pp. 37-41).

[4] R.E. Kalman, "Contributions to the theory of optimal control," Boletin De La Sociedad Matematica Mexicana, pp. 102-119. 1960.

[5] L.M. Silverman and B.D.O. Anderson, "Controllability, observability, and stability of linear systems," SIAM J. Control, Vol. 6, No. 1, pp. 121-130, 1966.

[6] M. Ikeda, H. Maeda, and S. Kodama, "Stabilization of linear systems," SIAM J. Control, Vol. 10, No. 4, 1972.

[7] B.D.O. Anderson and J.B. Moore, "New results in linear system stability," SIAM J. Contr., Vol. 7, No. 3, August 1969.

[8] W.H. Kwon and A.E. Pearson, "A modified quadratic cost problem and feedback stabilization of a linear system," IEEE Trans. Autom. Contr., Vol. AC-22,

No. 5, pp. 838-842, 1977.

[9] D.L. Kleinman, "On an iterative technique for Riccati Equation computations," IEEE Trans. Autom. Contr., pp. 114-115, Feb. 1968.

[10] P. Hartman, Ordinary Differential Equations, Birkhauser, 1982.

[11] J.K. Hale, Ordinary Differential Equations, Kreiger, 1980.

[12] J-P. Aubin, Applied Abstract Analysis, Wiley, 1977.

[13] H.L. Royden, Real Analysis, McMillan, 1968.

[14] J. Warga, Optimal Control of Differential and Functional Equations, Academic, 1972.

[15] E.J. Davison and E.G. Kunze, "Some sufficient conditions for the global and local controllability of nonlinear time-varying systens," SIAM J. Control, Vol. 8, pp. 489-494, 1970.

[16] L.M. Silverman and H.E. Meadows, "Controllability and observability in time-variable linear systems," SIAM J. Control, Vol. 5, No. 1, pp. 64-73, 1967.

[17] L. Rodman, "On extremal solutions of the algebraic Riccati equation," in Algebraic and Geometric Methods in Linear Systems Theory, AMS, Vol. 18, 1980, pp. 311-327.

[18] E. Emre and G. Knowles, "Stabilization of nonlinear time-varying systems: a systems over parameters approach," Aug. 1984, submitted.

[19] S. Bittanti, P. Colaneri, and G. Guardabassi, "Periodic solutions of Periodic Riccati equations," IEEE Trans. AC, July 1984.

5. PERIODIC SYSTEMS

In this section it is shown that application of the results in Section 2 to periodic systems leads to two important results. First, we show how a recent result of [19] can be obtained immediately also using the generalization of Kleinman's algorithm given in [18].

5.1. THEOREM [19]. *Let $\Sigma=(F,G,H)$ be a completely controllable and completely observable periodic system. Then the unique steady state solution of the time-varying Riccati equation (as defined in [4]) exists and is periodic.*

PROOF: It follows from [5] that Σ is uniformly completely controllable and uniformly observable. Thus, by [4], the unique steady state solution $\bar{P}$ exists. By [5], $W^{-1}(t,t+\sigma)$ is periodic, with the same period T_1 as Σ. It follows that each element of the sequence $V_0, V_1, \ldots$ constructed in [18] converging to $\bar{P}$, is periodic with T_1. Thus the limit of $\{V_i\}$,

$\bar{P}$ is periodic with T_1.□ It is known from [4] that $\bar{P}$ stabilizes (F,G).

The following results is a new result.

5.2. THEOREM. *Let T=a metric space. Let $\Sigma_\omega=(F_\omega,G_\omega)$ be a family of time-varying systems continuously parameterized by ω in T. Suppose that for each ω in T (F_ω,G_ω) is periodic for some period T_1 independent of ω. Suppose that (F_ω,G_ω) is completely controllable at every ω in T (see [4]). Then the steady-state solution of the Riccati equation $\bar{P}_\omega$ is a continuous function of ω in T, and is periodic with period T_1 for each ω in T.*

PROOF. By periodicity with T_1 on T, it follows from [5] that, there exists a constant $\sigma>0$ such that for each ω in T, (F_ω,G_ω) is uniformly completely controllable with the same σ. Since $W_\omega(\cdot,\cdot+\sigma)$ is periodic with T_1 for each ω in T, it is continuous with respect to ω with the uniform convergence topology on its codomain. As, also F_ω,G_ω are continuous in with uniform convergence topology on their codomains (due to periodicity with the same T_1), for each ω in T, there exists an open neighborhood U_ω of $\bar{\omega}$ on which (F_ω,G_ω) is uniformly bounded and equi-uniformly completely controllable for $t \geq 0$. Thus, by Theorem 2.9, $\bar{P}$ is continuous on U_ω for each ω in T. Thus, by corollary 2.10, it is continuous on T.□

5.3. REMARK. Clearly, if the family $\Sigma_\omega=(F_\omega,G_\omega)$ is a family of time-invariant systems, they satisfy Theorem 5.2. This is a result obtained previously (see, [3],[17]). Further implications of the results in [18] and here are currently being considered.

DELAY ANALYSIS OF THE n-ARY STACK ALGORITHM FOR A RANDOM-ACCESS BROADCAST CHANNEL

LAZAROS MERAKOS and CHATSCHIK BISDIKIAN
EECS Department
University of Connecticut
Storrs, Connecticut

ABSTRACT

We consider the random multiple accessing of a collision-type, time-slotted, packet-switched channel for a Poisson infinite-user model. We analyze an easy-to-implement, random access algorithm that allows users with newly arrived packets to freely access the channel independently of any collision resolution process that might be in progress. Furthermore, users sense the channel only while they have a packet to transmit. Utilizing the regenerative character of the stochastic processes that are associated with the random access system, we derive upper and lower bounds on the mean packet delay, and the packet delay variance induced by the algorithm.

I. INTRODUCTION

We consider the random access of a slotted channel, by asymptotically many identical, independent, and bursty, packet-transmitting users. The cumulative packet traffic is then Poisson, and the Poisson user model arises. We assume that each packet has length equal to one slot, that the transmission is synchronous (the transmission of a packet can start only at the beginning of some slot), and that feedback broadcast, per slot, exists. We consider the limited feedback sensing model, where the users observe the broadcasted feedback, only while they are blocked. Newly blocked users enter then the collision resolution process, immediately, and due to that, the limited feedback sensing protocols have last come-first serve characteristics, and they are more insensitive to channel errors, as compared to the full feedback sensing algorithms, in which users are required to monitor the feedback continuously, even when idle. This model was first considered by Tsybakov and Vvedenskaya [1], where ternary feedback ("empty" versus "success" versus "collision") was also assumed. In [2], Vvedenskaya and Tsybakov proposed a number of limited feedback sensing algorithms, for both binary ("collision" versus "no collision") and ternary feedback, and they studied the effect of channel errors to the performance of those algorithms. The maximum throughput attained by the algorithms in [2], is .401, for binary feedback, and it is .406, for ternary feedback. Some of these algorithms were independently developed and studied in [3], [4], [5]. In this study we treat only the algorithms that use binary feedback and n levels of splitting (n-ary stack algorithms). A more efficient stack algorithm has been recently introduced and analyzed in [7].

In [6] the packet delay characteristics of the n=2 algorithm have been studied through the use of a functional equation. In this paper we take a more direct approach, and we study the first two moments of the packet delay through the use of tight upper and lower bounds, for n=2, and n=3.

II. THE n-ARY STACK ALGORITHM FOR CNC BINARY FEEDBACK

Let time be measured in slot units, and let the integer k, denote then slot indices, where slot k occupies the transmission time interval, [k,k+1). Also, let F_k denote the binary feedback corresponding to slot k, where F_k= NC, and F_k=C represent respectively a noncollision versus collision slot, k. Let some packet arrive during slot k. We then assume that the packet attempts transmission during slot (k+1), and it observes the feedback, F_{k+1},

and all the feedbacks after that, until successfully transmitted. We also assume that the packet has no knowledge of the channel output history, F_i: $i<k$ (limited feedback sensing). In its effort to be transmitted successfully, the packet utilizes a parameter I_j, whose index corresponds to slot j, and it applies the following algorithm, which we name n-ary Stack Algorithm (SAn)

1. A packet arrived during slot (k-1), sets $I_k=1$
2. A packet attempts transmission within slot k, if and only if, $I_k=1$. The packet is successfully transmitted within slot k, if and only if, $I_k=1$ and $F_k=NC$.
3. The updating of the parameter I_k is done as follows.
   a) If $F_k=NC$, and $I_k>1$, then, $I_{k+1}=I_k-1$
   b) If $F_k=C$, and $I_k=1$, then, $I_{k+1}=J$; where J is an integer random variable uniformly distributed on $\{1,2,\ldots,n\}$, and n is an integer parameter, $n\geq 2$.
   c) If $F_k=C$, and $I_k>1$, then $I_{k+1}=I_k+n-1$

The operational characteristics of the n-ary Stack algorithm are perhaps better explained by introducing the concept of a "stack", as it was first done in [1]. A stack will be understood to mean an abstract storage device, consisting of an infinite number of cells, labelled 1,2,3,... The number of packets that a cell can accommodate is unrestricted. At the beginning of each slot, k, the n-th cell of the stack contains the packets with $I_k=n$. Packets are eventually successfully transmitted after moving through the cells of the stack in accordance with the rules defined above. To resolve conflicts, the algorithm splits uniformly the collided packets into the first n cells of the stack.

III. RENEWAL PROPERTIES, OUTPUT RATE, AND MEAN SESSION LENGTH

The network operates with the SAn in sessions. A session is a sequence of consecutive slots that begins and ends at two consecutive algorithm renewal instants to be defined below. These instants are denoted by R_n, $n\geq 1$, and are determined by means of a renewal counter. The first session begins at $R_1=1$, with the renewal counter set to "2". Depending on the channel outcomes, the renewal counter updates its indication in accordance with rules 3.a, and 3.c. The second renewal instant, R_2, is the instant at which the renewal counter drops to "1" for the first time; this signifies the end of the first session. Immediately after R_2 the renewal counter is reset to "2", and the second session begins. This process continues indefinitely.

A session starting with k packets in the first cell is called a session with multiplicity k; $k\geq 0$. Note that the renewal counter indication is always greater than the label number of every nonempty cell of the stack. Thus, when a session begins, all cells are empty except from cell #1, which is occupied by the k new packets that arrived during the last slot of the previous session.

The time from the instant that a session begins until it ends is called the length of the session. The session with multiplicity 0 (called the empty session), and the session with multiplicity 1 have length equal to unity. A session with multiplicity k, $k\geq 2$, has a random length that depends only on the arrival process of new packets during the session, and on the rules of the algorithm. Thus, in view of the independent and stationary increments property of the Poisson process that models the input traffic, the session lengths are independent, identically distributed (i.i.d.) random variables.

a. Output Rate

Let L_1, L_2, ... denote the lengths of successive sessions, measured in time units equal to the packet length. Then,

$$R_1 = 1 \; ; \; R_{i+1} = R_i + L_i \; , \; i=1,2,\ldots$$

define the algorithm renewal instants. The sequence $\{R_i\}_{i\geq 1}$ forms a delayed renewal process, since L_1, L_2, ... are i.i.d. non-negative variables.

Let

$$\mu(n) = \frac{1}{n} \sum_{j=1}^{n} I(\mathcal{O}_j)$$

where $I(\mathcal{O}_j)$ denotes the indicator functions of the event $\mathcal{O}_j \triangleq$ {successful transmission during the jth slot}. Thus, $\mu(n)$ represents the average fraction of time that successful packet transmissions have occurred on the channel by slot n.

Consider now an arbitrary session, say the ith, and let S_i denote the random number of packets that were successfully transmitted during the course of the session. Clearly, S_i depends on L_i, but the pairs (L_i,S_i), $i\geq 1$, are independent and identically distributed. Let $S=E(S_i)$, and $L=E(L_i)$. The expected number of successful transmissions, S, during a session can be thought of as an average reward earned during the session. With this in mind, we state the following result from the theory of renewal reward processes (see, for example, [8, sect. 3.6]).

Theorem 1 If $S<\infty$, and $L<\infty$, then there exists a real number, μ, such that $\lim_{n\to\infty} \mu(n) = \lim_{n\to\infty} E(\mu(n)) = \frac{S}{L} = \mu$ with probability 1

The above theorem states that the (expected) long-run average number of successful transmissions per unit time is just the expected number of successful transmissions during a session, divided by the mean session length, provided that both S and L are finite. The quantity μ is the channel's output rate.

Consider again the ith session; since at both R_i and R_{i+1} there are only newly arrived packets present in the system, it follows that the number, S_i, of successfully transmitted packets (if any) during the ith session is just the number of packets arrived at the system during the time interval $[R_i-1, R_{i+1}-1)$; all such packets will be called packets associated with session i, and their number is given by

$$S_i = \sum_{j=1}^{L_i} A_j \qquad (1)$$

where A_j denotes the number of arrivals in the interval $[R_n+(j-2),R_n+(j-1))$; $1\leq j\leq L_i$. Clearly, $\{A_j\}_{j\geq 1}$ is a sequence of independent Poisson (λ) random variables. Furthermore, L_i is a stopping time for $\{A_j\}_{j\geq 1}$ since the event $\{L_i=m\}$ is independent of $A_{m+1}, A_{m+2}, \ldots$. If we assume that $E(L_i)<\infty$, then, taking expectations in (2), and applying Wald's Lemma yields

$$S = E(S_i) = E(L_i)E(A_j) = \lambda L \qquad (2)$$

since $E(A_j) = \lambda$, and $E(L_i) = L$, for every i and j. In view of (2), Theorem 1 yields

Corollary 1: If $L<\infty$, then $\mu = \lambda$

Thus, the SAn maintains the rate if the mean session length is finite.

Next we investigate the conditions under which the mean and the variance of the session length are finite, and we establish bounds on them.

b. Mean and Variance of the Session Length

Consider a session of random multiplicity, $K\geq 0$, and let ℓ_K denote its random length. If we let $L_k=E(\ell_K|K=k)$, and $M_k=E(\ell_K^2|K=k)$, then the mean session length is given by

$$L = \sum_{k=0}^{\infty} p_k L_k \qquad (3)$$

and the session length variance, Q, is given by

$$Q = M - L^2 \tag{4}$$

where

$$M = \sum_{k=0}^{\infty} P_k M_k$$

and

$$P_k = e^{-\lambda} \lambda^k / k! \tag{5}$$

We proceed now with the investigation of the region of convergence of the series given in (3), and (4) by deriving and studying a system of equations for L_k, and M_k. The proofs of the results stated in this paper can be found in [10].
We first state the following

Proposition 1 The random length, ℓ_k, of a session with specified multiplicity, $k \geq 0$, satisfies the following system of equations.

$$\ell_o = \ell_1 = 1 \; ; \; \ell_k = 1 + \sum_{i=1}^{n} \ell_{k_i + N_i} \; ; \; k \geq 2 \tag{6}$$

; where $N_1, N_2, \ldots, N_n$ are independent, Poisson random variables with parameter λ, which are also independent of the random variables $k_1, k_2, \ldots, k_n$, which are multinomially distributed, with

$$P(K_1 = k_1, \ldots, K_n = k_n) = \frac{k!}{k_1! k_2! \ldots k_n!} \; n^{-k} \; ; \; 0 \leq k_i \leq k \; ; \; \sum_{i=1}^{n} k_i = k$$

In a straightforward way, Proposition 1 yields the following

Proposition 2
(i) The mean length, L_k, of a session with specified multiplicity k, $k \geq 0$, satisfies the following system of equations:

$$x_k = \sum_{i=2}^{\infty} a_{k,i} \, x_i + c_k \; , \; k \geq 2 \tag{7}$$

where[1]

$$a_{k,i} = n(p_i * b_i(k, n^{-1})), \; i \geq 0, \; k \geq 0; \; b_i(k,p) \triangleq \binom{k}{i} p^i (1-p)^{k-i}, k \geq 2, 0 \leq p \leq 1;$$

p_i, $i \geq 0$, is as given in (5); $c_k = 1 + a_{k,0} + a_{k,1}$, $k \geq 2$.

(ii) The mean square length, M_k, of a session with specified multiplicity k, $k \geq 2$, satisfies the following system of equations

$$x_k = \sum_{i=2}^{\infty} a_{k,i} \, x_i + d_k \; ; \; k \geq 2 \tag{8}$$

where

$$d_k = d_k(\{L_i, i \geq 0\}) = 2(L_k - 1) + n(n-1) \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \sum_{k_1=0}^{k} \sum_{k_2=0}^{k-k_1} p_i p_j \, b_k(k_1, k_2) \, L_{k_1 + j} \, L_{k_2 + i}$$

$$b_k(k_1, k_2) = \frac{k!}{k_1! k_2! (k - k_1 - k_2)!} \; n^{-k} (n-2)^{k - k_1 - k_2} \; ; \; 0 \leq k_1 + k_2 \leq k$$

and the coefficients $a_{k,i}$ are as defined in part (i).

Formally, both system (7) and system (8) always have an "infinite" solution $x_o = x_1 = 1$, $x_k = \infty$, $k \geq 2$. The following theorem specifies a sufficient condition under which both systems have a non-negative, finite solution.

Theorem 2
(i) Given n; then, for every $\lambda < \lambda_o^{(n)}$, system (7) has a solution $\{y_k; k \geq 2\}$,

(1) * denotes convolution

and system (8) has a solution $\{z_k; k\geq 2\}$, such that

$$0 \leq b\ k-c \leq y_k \leq b\ k-c \quad ; \ k \geq 2$$

and

$$0 \leq \alpha_2' k^2 + \alpha_1' k + \alpha_o' \leq z_k \leq \alpha_2 k^2 + \alpha_1 k + \alpha_o \ ; \ k \geq 2$$

; where

$$\lambda_o(n)=((8n^3-7n^2+2n+1)^{1/2}-3n+1)/(2n(n-1))$$

; where b, c, b', c', α_2, α_1, α_o, α_2', α_1', α' are bounded functions of λ, and n. (Their expressions can be found in [10]).

(ii) Given n, then, for $\lambda<\lambda_o(n)$, $L_o=L_1=1$, $L_k=y_k$, $k\geq 2$, and $M_o=M_1=1$, $M_k=z_k$; $k\geq 2$.

Given n, let $\lambda\in[0,\lambda_o(n))$; then, in view of (3), and of Theorem 2 we have

$$0 \leq B(b',c') \equiv L_\ell \leq L \leq L_u \equiv B(b,c) \tag{9}$$

; where,

$$B(x_1,x_o) \triangleq \lambda(1-e^{-\lambda})x_1 - (1-e^{-\lambda}-\lambda e^{-\lambda})x_o + (1+\lambda)e^{-\lambda}$$

$$0\leq Q_1(\alpha_2',\alpha_1',\alpha_o') \equiv M_\ell \leq M \leq M_u \equiv Q_1(\alpha_2,\alpha_1,\alpha_o)<\infty \tag{10}$$

; where $Q_1(y_2,y_1,y_o) \triangleq \lambda(1+\lambda-e^{-\lambda})y_2+\lambda(1-e^{-\lambda})y_1+(1-e^{-\lambda}-\lambda e^{-\lambda})y_o+(1+\lambda)e^{-\lambda}$

From (9) and Corollary 1 we have the following.

Corollary 2 The SAn maintains the rate for every $\lambda\in[0,\lambda_o(n))$

The values of $\lambda_o(n)$ for n=2,3, and 4, as obtained from (4) are as follows: $\lambda_o(2)=0.3507$, $\lambda_o(3)=0.3874$, and $\lambda_o(4)=0.3843$. For $n\geq 4$, $\lambda_o(n)$ is monotonically decreasing.

IV MEAN AND VARIANCE OF PACKET DELAY

Consider the random access system operating with the SAn over the time interval $[0,+\infty)$. Packets arrive at the network at time instants a_n, n=1,2,..., where $0\leq a_1\leq a_2\leq \ldots$. Let the arriving packets be labelled n=1,2,... in accordance to their arrival instant. We define the delay, $\mathcal{D}_n$, experienced by the nth packet as the time difference between its arrival at the transmitter and the end of its successful transmission (so that $\mathcal{D}_n=1$, when the packet is successfully transmitted beginning at the same moment it arrives at the transmitter). Let the random variable N_i denote the total number of packets associated with (i.e. arrived and successfully transmitted during) the first i non-empty sessions. Let, also, $\bar{S}_i$ denote the number of packets associated with the ith non-empty session. We have that $N_0=0$; $N_{i+1}=N_i+\bar{S}_{i+1}$, i=0,1,2,... The sequence $\{N_i\}_{i>0}$ is a renewal process, since $\{\bar{S}_i\}_{i>1}$ is a sequence of i.i.d. random variables. Furthermore, the renewal properties of the algorithm clearly indicate that, whenever a non-empty session ends, the arrival and waiting-time mechanisms are "reset" by the next arrival; thus, the process $\{\mathcal{D}_{N_i+n}\}_{n\geq 1}$, for every $i\geq 0$, is a probabilistic replica of the process $\{\mathcal{D}_n\}_{n\geq 1}$. Thus, the discrete-time process $\{\mathcal{D}_n\}_{n>1}$ is regenerative respective to the imbedded renewal process $\{N_i\}_{i>0}$, with common regenerative cycle, $\bar{S}$, the number of packets associated with a non-empty session.

The next theorem is similar to Theorem 1 and depends on the theory of regenerative processes (see [9, Thm 2]).

Theorem 3 If (A.1) $\bar{S}$ is not periodic, with $\bar{S}\triangleq E(\bar{S})<\infty$, and (A.2) $\bar{T}\triangleq E(\sum_{i=1}^{\bar{S}} \mathcal{D}_i)<\infty$ then there exists a parameter D such that

$$D = \lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \mathcal{D}_n = \lim_{n\to\infty} n^{-1} E(\sum_{i=1}^{n} \mathcal{D}_n) \quad \text{with probability 1.}$$

Furthermore, $\mathcal{D}_n$ converges in distribution to a random variable $\mathcal{D}_\infty$, and

$$D = E(\mathcal{D}_\infty) = \bar{T}/\bar{S} < \infty$$

Thus, under assumptions (A.1) and (A.2), the limiting average, the limiting expected average, and the mean of the limiting distribution of $\{\mathcal{D}_n\}_{n\geq 1}$, exist, coincide, and are finite; their common value, D, will be referred to as the mean packet delay induced by the SAn. Next we elaborate on the assumptions stated in Theorem 3. From the operation of the SAn it can be easily seen that $\bar{S}$ is not periodic. Let S_k denote the expected number of packets associated with a session of multiplicity $k\geq 0$. Then, noting that $S_0=0$, we have

$$\bar{S} = \frac{1}{1-p_0} \sum_{k=1}^{\infty} p_k S_k = \frac{1}{1-p_0} \sum_{k=0}^{\infty} p_k S_k = \frac{1}{1-p_0} S \qquad (11)$$

Given n, let $\lambda\in[0,\lambda_o(n))$; then, from (2), (11), and Theorem 2 , we have that

$$\bar{S} = \lambda L/(1-e^{-\lambda}) < \infty \qquad (12)$$

Thus, assumption (A.1) is true, if $\lambda<\lambda_o(n)$. We proceed now to show that this is also the case for assumption (A.2).
Consider the nth packet arrival; its delay can be expressed as

$$\mathcal{D}_n = A_n + C_n$$

A_n denotes the nth packet access waiting time from the packet arrival instant to the instant the packet enters the stack for the first time.

C_n denotes the nth packet contention waiting time from the instant the packet enters the stack for the first time till the end of its successful transmission.

Using the above decomposition of the packet delay, we write the mean cumulative delay, $\bar{T}$, experienced by all packets associated with a non-empty session as

$$\bar{T} = U + V \qquad (13)$$

where $U \triangleq E(\sum_{i=1}^{\bar{S}} A_i)$, and $V \triangleq E(\sum_{i=1}^{\bar{S}} C_i)$ represent the mean cumulative access and contention delay, respectively, over a non-empty session. Clearly, $A_i<1$, and therefore, $U<\bar{S}<\infty$, for every $\lambda<\lambda_o(n)$. Now consider the mean cumulative contention delay, V, and let V_k denote the cumulative contention delay over a session of multiplicity k; $k=0,1,2,\ldots$. The rules of the algorithm yield the following relation for V_k:

Proposition 3
$V_0=0$, $V_1=1$

$$V_k = k + \sum_{i=1}^{n-1} \bar{K}_i \ell_{K_i} + N_i + \sum_{i=1}^{n} V_{K_i+N_i} \qquad k \geq 2 \qquad (14)$$

where N_i, K_i, $1\leq i\leq n$, are as defined in Proposition 1, and

$$\bar{K}_i = \sum_{j=i+1}^{n} K_j \ , \quad 1 \leq i \leq n-1$$

Taking expectations in both sides of (14) yields the following

Proposition 4
The conditional mean cumulative delay, $V_k \triangleq E(V_k)$, of a session with multiplicity k, $k\geq 2$, satisfies the following system of equations:

$$x_k = \sum_{i=2}^{\infty} a_{k,i}\, x_i + f_k \; ; \; k \geq 2 \qquad (15)$$

where

$$f_k = f_k(\{L_i; i\geq 2\}) = k + a_{k,i} + \sum_{\xi=0}^{n-2} \sum_{j=0}^{k} \sum_{m=0}^{\infty} \sum_{i=0}^{k-j} b_j(k,\frac{\xi}{n}) b_i(k-j,\frac{1}{n-\xi}) P_m(k-j-i) L_{i+m},$$

and where $a_{k,i}$, $b_i(\cdot,\cdot)$ and p_i are as defined in Proposition 2.

The following is a result analogous to Theorem 2.

Theorem 4

(i) Given n; then, for every $\lambda<\lambda_o(n)$, system (15) has a solution $\{v_k; k\geq 2\}$ such that

$$0 < \beta'_2 k^2 + \beta'_1 k + \beta'_o \leq v_k \leq \beta_2 k^2 + \beta_1 k + \beta_o$$

; where β_2, β_1, β_o, β'_2, β'_1, β'_o are functions of λ and n. (Their expressions can be found in [10]).

(ii) Given n; then, for $\lambda<\lambda_o(n)$, $V_o=0$, $V_1=1$, $V_k=v_k$, $k \geq 2$.
The mean cumulative contention delay, V, over a nonempty session is given by

$$V = (1-p_o)^{-1} \sum_{k=1}^{\infty} p_k V_k$$

From Theorem 4 we have that, if $\lambda<\lambda_o(n)$, then

$$0 \leq Q_2(\beta'_2,\beta'_1,\beta'_o) \equiv V_\ell \leq V \leq V_u = Q_2(\beta_2,\beta_1,\beta_o) < \infty \qquad (16)$$

; where

$$Q_2(x_2,x_1,x_o) = \frac{1}{1-p_o}(p_1 + \sum_{k=2} p_k(x_2 k^2 + x_1 k + x_o)) = (1+\frac{\lambda}{1-e^{-\lambda}})x_2 + \lambda x_1 + (1-\frac{\lambda e^{-\lambda}}{1-e^{-\lambda}})x_o + \frac{\lambda e^{-\lambda}}{1-e^{-\lambda}}$$

Finally in view of (13), and (16), we have the following

Corollary 3

If $\lambda<\lambda_o(n)$, then both assumptions in Theorem 3 hold; thus, the mean packet delay, D(λ), induced by SAn is finite.

Definition: A random-access algorithm is called stable if the mean packet delay is finite. The maximum stable throughput, η, of a random-access algorithm is defined as the supremum of the cumulative input rate, λ, so that the algorithm is stable; that is, $\eta \triangleq \sup(\lambda: D(\lambda)<\infty)$.

Given n, let η(n) denote the maximum stable throughput of the SAn; then, from Corollary 3, we have $\eta(n)\geq\lambda_o(n)$, since $\lambda<\lambda_o(n)$ is only a sufficient condition for finite mean packet delay.

a. Bounds on the mean packet delay

Given n, and $\lambda<\lambda_o(n)$, then for Theorem 3 and (13) we have

$$D = \bar{T}/\bar{S} = A + C \qquad (17)$$

where A, and C are the mean access delay and the mean contention delay, respectively.

Consider first the mean access delay, A. Clearly, the access delay of the ith packet, A_i, is uniformly distributed over the unit interval, for every $i\geq 1$, since the arrival process is Poisson. Thus,

$$A = E(A_i) = \frac{1}{2} \; ; \; i \geq 1$$

Next consider the mean contention delay, C. Given n, and $\lambda<\lambda_o(n)$, we have $C=V/\bar{S}$, where $\bar{S}$ is as given in (12). Thus

$$C = (1-e^{-\lambda})\, V/(\lambda L) \qquad (18)$$

Using the bounds on V, and L, as given by (9) and (16), respectively, in (18), and (17) yields

$$\frac{1}{2} + (1-e^{-\lambda})V_\ell/(\lambda\, L_u) \leq D \leq \frac{1}{2} + (1-e^{-\lambda})V_u/(\lambda\, L_\ell) \qquad (19)$$

The above bounds are tight enough for small values of λ, but they become loose as λ approaches $\lambda_o(n)$. In the rest of this section we develop a method for computing tighter bounds on L, M, and D.

Given some finite natural number, $N \geq 2$, let us consider the following system of N linear equations:

$$x_k = \sum_{i=2}^{N+1} a_{k,i}\, x_i + b_k \; ; \; 2 \leq k \leq N+1 \qquad (20)$$

; where b_k, $2 \leq k \leq N+1$, non-negative real constants, and $a_{k,i}$, $2 \leq i \leq N+1$, are as defined in Proposition 2.

Lemma 1

Given n, let $\lambda<\lambda_o(n)$; then, the matrix $(I_N-A_N)^{-1}$ exists and has non-negative elements for every $N \geq 2$; where

$A_N=(a_{ij})$ is the (NxN) non-negative, square matrix with $a_{ij}=a_{i,j}$, $2 \leq i \leq N+1$, $2 \leq j \leq +1$, and I_N is the (NxN) unit matrix.

Corollary 4

System (20) has a unique non-negative solution, for every $b_k \geq 0, 2 \leq k \leq N+1$, for every $N \geq 2$.

Theorem 5

Given n, let $\lambda<\lambda_o(n)$; then, for any $N \geq 2$

$$L_k^\ell \leq L_k \leq L_k^u \; ; \; 2 \leq k \leq N+1$$
$$M_k^\ell \leq M_k \leq M_k^u \; ; \; 2 \leq k \leq N+1$$
$$V_k^\ell \leq V_k \leq V_k^u \; ; \; 2 \leq k \leq N+1$$

; where $\{L_k^u \; ; \; 2 \leq k \leq N+1\}$, $\{L_k^\ell \; ; \; 2 \leq k \leq N+1\}$, $\{M_k^u \; ; \; 2 \leq k \leq N+1\}$, $\{M_k^\ell \; ; \; 2 \leq k \leq N+1\}$, $\{V_k^u \; ; \; 2 \leq k \leq N+1\}$, and $\{V_k^\ell \; ; \; 2 \leq k \leq N+1\}$ are the unique solutions to system (20) with

$b_k = \sum_{i=N+2}^{\infty} a_{k,i}(bi-c)+c_k$, $b_k = \sum_{i=N+2}^{\infty} a_{k,i}(b'i-c')+c_k$,

$b_k = \sum_{i=N+2}^{\infty} a_{k,i}(\alpha_2 i^2+\alpha_1 i+\alpha_o)+d_k(\{L_i, i \geq 0\})$, $b_k = \sum_{i=N+2}^{\infty} a_{k,i}(\alpha_2' i^2+\alpha_1' i+\alpha_o')+d_k(\{G_i, i \geq 0\})$

$b_k = \sum_{i=N+2}^{\infty} a_{k,i}(\beta_2 i^2+\beta_1 i+\beta_o)+f_k(\{F_i, i \geq 0\})$, and $b_k = \sum_{i=N+2}^{\infty} a_{k,i}(\beta_2' i^2+\beta_1' i+\beta_o')+$

$+f_k(\{G_i, i \geq 0\})$,

respectively; where $F_o=F_1=G_o=G_1=1$; $F_i=L_i^u$, $2 \leq i \leq N+1$; $F_i=bi-c$, $i \geq N+2$; $G_i=L_i^\ell$, $2 \leq i \leq N+1$; $G_i=b'i-c'$, $i \geq N+2$; where $a_{k,i}, c_k, d_k(\cdot), b, c, b', c', \alpha_2, \alpha_1, \alpha_o, \alpha_2', \alpha_1'$, and α_o' are as given in Proposition 2, and where $f_k(\cdot), \beta_2, \beta_1, \beta_o, \beta_2', \beta_1'$, and β_o' are as given in Proposition 4.

Combining the results of Theorem 5 with those of Theorem 2 yields the following bounds:

$$L_\ell^N \leq L \leq L_u^N \; ; \; M_\ell^N \leq M \leq M_u^N \; ; \; V_\ell^N \leq V \leq V_u^N \; ; \; \lambda < \lambda_o(n)$$

; where

$$L_u^N = L_u - \sum_{k=2}^{N+1} p_k(b\,k-c - L_k^u(\lambda)) \qquad L_\ell^N = L_\ell + \sum_{k=2}^{N+1} p_k(L_k^\ell - b'k-c')$$

$$M_u^N = M_u - \sum_{k=2}^{N+1} p_k(\alpha_1 k^2+\alpha_1 k+\alpha_o - M_k^u) \qquad M_\ell^N = M_\ell + \sum_{k=2}^{N+1} p_k(M_k^\ell - \alpha_2' k^2 - \alpha_1' k - \alpha_o)$$

$$V_u^N = V_u + \sum_{k=2}^{N+1} p_k(V_k^u-\beta_2 k^2-\beta_1 k-\beta_o) \qquad V_\ell^N = V_\ell - \sum_{k=2}^{N+1} p_k(\beta_2' k^2+\beta_1' k+\beta_o'-V_k^\ell)$$

; where (L_u,L_ℓ), (M_u,M_ℓ), and (V_u,V_ℓ) are as given in (9), (10), and (16), respectively.

Finally, using the above bounds in (4), and (19) yields

$$Q_\ell^N \triangleq M_\ell^N - (L_u^N)^2 \le Q \le M_u^N - (L_\ell^N)^2 \triangleq Q_u^N$$

$$Q_\ell^N \triangleq \frac{1}{2} + (1-e^{-\lambda})V_\ell^N/(\lambda\, L_u^N) \le D \le \frac{1}{2} + (1-e^{-\lambda})V_u^N/(\lambda\, L_\ell^N) \triangleq D_u^N$$

Table 1 gives the bounds L_ℓ^N, L_u^N on the mean session length, the bounds Q_u^N, Q_ℓ^N on the session length variance, and the bounds D_u^N, D_ℓ^N on the mean packet delay induced by the SA2,and the SA3, for N=15, and $\lambda<\lambda_o(n)$. Note that the bounds remain extremely tight even for λ close to $\lambda_o(n)$.

To evaluate the packet delay variance we combined a technique developed in [6] with the bounding techniques described above. The following bounds on the packet delay variance, H, are adopted from [10].

$$H_\ell^N \le H \le H_u^N \;;\; \lambda < \lambda_o(n)$$

; where N is the dimension of a linear system of equations, the solution of which was used to evaluate the bounds (see [10]). The bounds H_ℓ^{15}, H_u^{15} on the variance of the packet delay induced by SA2, and SA3 are included in Table 1.

REFERENCES

[1] B.S. Tsybakov and N.D. Vvedenskaya, "Random Multiple-Access Stack Algorithm," Probl. Peredachi Inf., Vol. 16, No. 3, pp. 80-94, 1980.

[2] N.D. Vvedenskaya and B.S. Tsybakov, "Random Multiple Access of Packets to a Channel with Errors," Probl. Peredachi Inf., Vol. 19, No. 2, pp. 52-68, 1982.

[3] L. Merakos, "Limited Sensing Random Multiple Access Using Binary Feedback," 1983 Johns Hopkins Conf. on Information Sciences and Systems.

[4] P. Mathys, and P. Flajolet, "Q-ary Collision Resolution Algorithms in Random-Access Systems with Free or Blocked Channel Access," in Abst. Papers, 1983 IEEE Inform. Theory Symp., St. Jovite, Quebec, Canada, Sept. 1983.

[5] G. Fayolle, P. Flajolet and M. Hofri, "On a Functional Equation Arising in the Analysis of a Protocol for a Multi-Access Broadcast Channel," Int. Rept., INRIA, France, April 1982.

[6] G. Fayolle, P. Flajolet, M. Hofri, and P. Jacquet, "The Evaluation of Packet Transmission Characteristics in a Multi-Access Collision Detection Channel with Stack Collision Resolution Protocol," Technion, Computer Science Dept., Technical Report #293, Oct. 1983.

[7] L. Georgiadis, and P. Papantoni-Kazakos, "Limited Feedback Sensing Algorithm for the Broadcast Channel," University of Connecticut, EECS Dept., Technical Report #TR-84-8, June 1984.

[8] S. Stidham, Jr., "Regenerative Processes in the Theory of Queues, with Applications to the Alternating-Priority Queue," Adv. Appl. Probl., Vol. 4, pp. 542-577, 1972.

[9] F. J. Beutler, "Mean Sojourn Times in Markow Queueing Networks: Little's

Formula Revisited," IEEE Trans. Inform. Theory, Vol. IT-29, no. 2, pp. 233-241, March 1983.

[10] C. Bisdikian, and L. Merakos, "Delay Analysis of the n-ary Stack Algorithm for a Random Access Broadcast Channel," University of Connecticut, EECS Dept., Technical Report #TR-84-10, Sept. 1984.

Table I. Upper and lower bounds on the mean, L, and the variance, Q, of the session length and on the mean, D, and variance, H, of the packet delay, for the SA2 and the SA3

	λ	0.05	0.10	0.15	0.20	0.25	0.30	0.35
n=2	$L_u^{15} \tilde{=} L_\ell^{15}$	1.00565	1.02622	1.07113	1.16167	1.35801	1.92055	8.22892
	$Q_u^{15} \tilde{=} Q_\ell^{15}$	0.04301	0.26668	1.04051	3.81549	16.69959	134.3063	34315.62
	$D_u^{15} \tilde{=} D_\ell^{15}$	1.68404	1.96913	2.44552	3.33212	5.29176	11.38280	87.28995
	$H_u^{15} \tilde{=} H_\ell^{15}$	1.14196	3.61181	10.21338	32.06220	136.0813	1202.678	395953.3
n=3	$L_u^{15} \tilde{=} L_\ell^{15}$	1.00619	1.02788	1.07265	1.15587	1.31346	1.65242	2.69508
	$Q_u^{15} \tilde{=} Q_\ell^{15}$	0.04578	0.26393	0.93019	2.92122	9.74869	42.32418	398.7818
	$D_u^{15} \tilde{=} D_\ell^{15}$	1.67669	1.92764	2.30451	2.91389	4.01133	6.37571	13.87780
	$H_u^{15} \tilde{=} H_\ell^{15}$	1.00399	2.89663	7.23018	18.76010	57.26140	247.4444	2517.139

THE DISTRIBUTION OF DELAY AND CAUDAL CHARACTERISTIC CURVE OF CSMA/CD NETWORKS

STEVEN L. BEUERMAN, EDWARD J. COYLE
Purdue University, West Lafayette, Indiana

ABSTRACT

A continuous time Markov chain model is developed which allows the calculation of the variance and distribution of the delay encountered by a packet being transmitted over a CSMA/CD network. Also, the caudal characteristic curves of the CSMA/CD network are computed. These curves provide a qualitative description of the performance of the network. Lastly, throughput curves are provided for the network when the expected delay is bounded. In this paper, all network parameters are assumed to be exponentially distributed, but this is only for ease of presentation.

I. INTRODUCTION

The Carrier Sense Multiple Access protocol with Collision Detection (CSMA/CD) provides high channel utilization and low expected delay for packet communication networks in which the average packet length is large compared with the maximum propagation delay across the network. Since many local area computer networks and packet radio networks satisfy this condition, the CSMA protocol has become popular with the designers of these networks. In recent years, it has appeared in several commercially available cable networks [1,2] and has been proposed for future mobile communication systems. Its popularity will continue to rise as the need for communication systems of this type increases.

As the popularity of CSMA/CD networks grows, network designers will need more information before they can build a network that suits their needs. For example, if the data to be sent over the network is delay sensitive, the designer will need to know more than just the expected delay. Variance of the delay encountered by a packet on the network, and the maximum throughput that can be achieved with a given bound on the delay are two statistics he would need. Both of these statistics are provided in this paper.

Several methods have been used to analyze CSMA networks [3-6]. Recently, a continuous time Markov Chain model was developed for an asynchronous, nonpersistent CSMA/CD network that allowed their complete characterization by a matrix whose dimensions are independent of the number of users in the network [7-8]. This model has already been used to find the performance of finite population networks and to obtain many analytical results when an infinite population and a constant load transmission are considered. These analytical results include stability conditions for the network, distribution of the number of busy users, an exact closed form expression for the expected delay, and the exact upper bound on the throughput.

In this paper, we use this model to obtain maximum throughput curves for a given expected delay. A revised version of this model, developed in section II, allows the calculation of the variance and distribution of delay. These results are discussed in Sections IV and V, respectively. In section III, the caudal characteristic curves of the CSMA/CD network are presented.

II. THE CSMA/CD MODEL

In [8], a matrix-geometric model was developed for CSMA/CD networks. The results which follow depend heavily on this model. We therefore provide a short summary of the important ideas in this model. The reader should see [8] for more details. It should also be noted that, in this paper, all network parameters, that is the packet length, transmission detection time, and conflict truncation time, are assumed to be exponentially distributed. While this is not a situation that is likely to be encountered in an operating network, it serves to demonstrate that the following data may be obtained. The technique is altered only slightly to account for arbitrary distributions of these parameters.

The most important concepts of the model are those of the busy user and the channel states for the network. A user is defined to be busy from the moment it receives a new packet for transmission until the moment after that same packet is successfully transmitted. Otherwise, the user is said to be idle. The number of busy users should be kept to a minimum. The state transition diagram for the model in [8] is shown in Figure 1. Each block in the diagram is a distinct channel state in which n users are busy. All transitions into these states are shown, along with their rate. The channel states are:

- 0 The 0 state is the idle state. No users are transmitting in the channel.
- 1 The 1 state occurs when one user is transmitting but has not yet been heard by all of the other users.
- D The D or departed state represents a packet whose transmission has been completed before the detection time has expired.
- P The P state is the safe transmission state in which the user's transmission cannot be interfered with.
- C The C, or collision state, occurs when two or more users are transmitting simultaneously.

The parameters of the network (see [8] for details) are as follows:

A) The packet length is an exponentially distributed random variable with mean $1/\lambda_p$.

B) The detection time is exponentially distributed with mean $1/\lambda_d$.

C) The collision truncation time is exponentially distributed with mean $1/\lambda_c$.

D) Each user receives packets at rate λ, with exponential interarrival times of mean length $1/\lambda$. When n users are busy, the rate at which the remaining (N-n) users generate new packets is $G(n) = (N-n)\lambda$.

E) If a user's transmission is delayed, the delay is exponentially distributed with mean $1/R(n)$ where n is the current number of busy users. The total rate of transmission attempts is $T(n) = G(n) + nR(n)$. The quantity $nR(n)$ is a constant and is denoted by γ.

In order to find the distribution of the delay a packet suffers before being transmitted, the model of [8] must be modified to allow for the tracking of an individual "tagged" packet. Thus, a revised model was developed. With these modifications, the set of possible states when there are i busy users, not including the user being tracked becomes

$$N(i) = \left[00 \;\; 10 \;\; D0 \;\; P0 \;\; C0 \;\; 01 \;\; 0D \;\; CC \;\; 0P \right]_i.$$

The first index in each state represents the activity of the network on the channel. The second index represents the interaction of the user being tracked, with the channel. For example, when the user with the tagged packet is idle and another user has transmitted a packet that has not yet been detected by the other users, the channel is said to be in state 10_i, where the subscript i denotes the number of busy users. When the tagged packet has collided with another packet on the channel, the

channel is in state CC_i. As a demonstration of the activity of this system, $00_i \rightarrow 10_i \rightarrow P0_i \rightarrow 00_{i-1}$ represents a successful transmission of a packet other than the tagged packet. $00_i \rightarrow 10_i \rightarrow CC_i \rightarrow 00_i$ represents the attempt of the user being tracked to transmit its packet. However, in the process, it hits a packet currently on the channel so the transmission is unsuccessful.

A note on boundary conditions: for i=1, the state C0 is impossible, and for i=0, only the four states which have a 0 as their first index are allowed.

There is still one more state in the state space. It is a state we call z, which is the absorbing state corresponding to the successful transmission of the tagged packet across the network. An example of a transition to this state would be $00_i \rightarrow 01_i \rightarrow 0P_i \rightarrow z$ which represents the successful transmission of the tagged packet. Hence, with this model, the delay of the tagged packet is identical to the time to absorption of the continuous time Markov chain model. The state space is therefore

$$\{ z, \mathbf{N}(0), \mathbf{N}(1), \mathbf{N}(2), \cdots \}.$$

Let t be the arrival time of the tagged packet. The network up to this time t has states with probabilities determined by the old model [8]. When this packet arrives, the state space is changed to the modified network. The initial probabilities of this new network are determined as follows. If the network is in the protected state when the tagged packet arrives, the initial state of the modified network would be $P0_i$. If the network is idle, the initial state would be 01_i. Otherwise, the modified network will start in the CC_i state, since the new packet will collide with another packet that is currently being transmitted. The initial probability vector will be called $\boldsymbol{\alpha}$.

The order in which packets are transmitted is random. When a packet is retransmitted, the probability that it is the tagged packet is $1/(i+1)$.

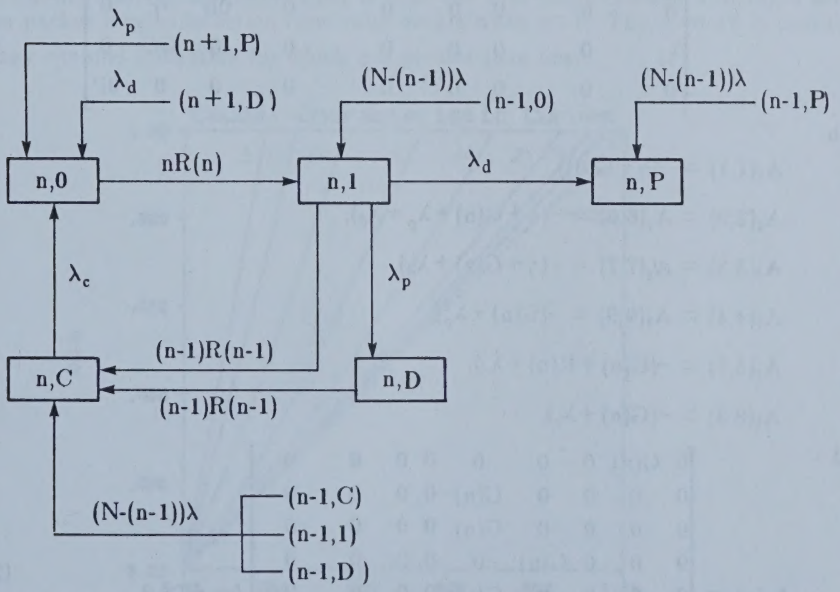


Figure 1: State transistion diagram for the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol developed in [8]. The simplest case is shown - all distributions are exponential. All possible transitions into and between states with n busy users are shown along with the rate at which each transistion occurs.

The generator matrix of this Markov Chain, has the form:

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}(0) & \mathbf{A}_1(0) & \mathbf{A}_2(0) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}(1) & \mathbf{A}_0(1) & \mathbf{A}_1(1) & \mathbf{A}_2(1) & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}(2) & \mathbf{0} & \mathbf{A}_0(2) & \mathbf{A}_1(2) & \mathbf{A}_2(2) & \mathbf{0} & \cdots \\ \mathbf{B}(3) & \mathbf{0} & \mathbf{0} & \mathbf{A}_0(3) & \mathbf{A}_1(3) & \mathbf{A}_2(3) & \cdots \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{T}^0 & \mathbf{T} \end{bmatrix} \qquad (2.1)$$

where

$$\mathbf{B}(n) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \lambda_d & 0 & \lambda_p \end{bmatrix}^T \text{ for } n \geq 2, \qquad (2.2)$$

$$\mathbf{A}_1(n) = \begin{bmatrix} 00 & \gamma\text{–}R(n+1) & 0 & 0 & 0 & R(n+1) & 0 & 0 & 0 \\ 0 & 10 & \lambda_p & \lambda_d & \gamma\text{–}R(n) & 0 & 0 & R(n) & 0 \\ 0 & 0 & D0 & 0 & \gamma\text{–}R(n) & 0 & 0 & R(n) & 0 \\ 0 & 0 & 0 & P0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_c & 0 & 0 & 0 & C0 & 0 & 0 & R(n) & 0 \\ 0 & 0 & 0 & 0 & 0 & 01 & \lambda_p & \gamma & \lambda_d \\ 0 & 0 & 0 & 0 & 0 & 0 & 0D & \gamma & 0 \\ \lambda_c & 0 & 0 & 0 & 0 & 0 & 0 & CC & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0P \end{bmatrix} \text{ for } n \geq 2, \qquad (2.3)$$

with

$$A_1(1,1) = -(\gamma + G(n)),$$

$$A_1(2,2) = A_1(6,6) = -(\gamma + G(n) + \lambda_p + \lambda_d),$$

$$A_1(3,3) = A_1(7,7) = -(\gamma + G(n) + \lambda_d),$$

$$A_1(4,4) = A_1(9,9) = -(G(n) + \lambda_p),$$

$$A_1(5,5) = -(G(n) + R(n) + \lambda_c),$$

$$A_1(8,8) = -(G(n) + \lambda_c),$$

and

$$\mathbf{A}_2(n) = \begin{bmatrix} 0 & G(n) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & G(n) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & G(n) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & G(n) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & G(n) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & G(n) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & G(n) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & G(n) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G(n) \end{bmatrix} \text{ for } n \geq 2. \qquad (2.4)$$

$\mathbf{A}_0$ (n) has only two entries. The first corresponds to a transition from $D0_n \rightarrow 00_{n-1}$ and occurs at rate λ_d, and the second corresponds to a transition from $P0_n \rightarrow 00_{n-1}$ which occurs at rate λ_p. For $n<2$, the rows and columns corresponding to nonexistent states must be deleted and the diagonal term adjusted accordingly, so that the sum of all terms in a row of A is 0.

III. CAUDAL CHARACTERISTIC CURVE

The caudal characteristic curve is a tool used to qualitatively describe the behavior of a queueing system [9]. This curve is a graph of η as a function of ρ where η is the spectral radius of the **R** matrix corresponding to the matrix geometric model in [8] and ρ is the traffic intensity of the model. For an M/M/1 queue, the caudal characteristic curve is just a straight line since $\eta=\rho$ for $0\leq\rho\leq1$. A higher value of η for a given ρ indicates that the queue is performing worse than an M/M/1 queue. For high values of η, the queue exhibits large and long-lived excursions. These excursions may be rare, but when they occur, the queue length is affected for a long period of time.

The caudal characteristic curve for the CSMA/CD network model are shown in Figure 2, in which each curve represents a different packet length/detection time ratio. For small ratios, the network behaves rather poorly. As the detection time decreases, however, the network starts to resemble an M/M/1 queue. This makes intuitive sense since when this ratio is very large, the probability of a collision is very low. An arriving packet would then see only the packets "ahead" of it, being served one after another, each of them requiring an exponentially distributed amount of time to be served. Thus, the network, when the packet length/detection time ratio is large, behaves like an M/M/1 queue with random service order. From Figure 2, this ratio must be at least 50 to obtain this type of behavior.

Another interesting observation is that the maximum channel utilization for a given packet length/detection time ratio occurs when $\eta=1$. The network is unstable for any channel utilization for which η is greater than one.

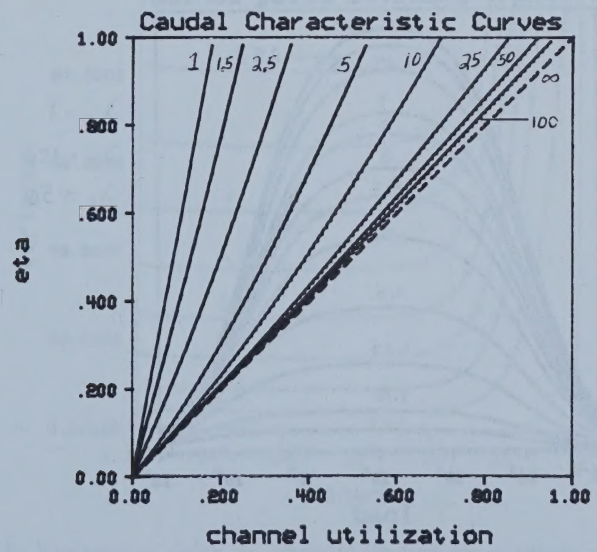


Figure 2: Caudal characteristic curve for the CSMA/CD protocol operating under various (packet length)/(detection time) ratios.

IV. CONSTANT DELAY AND VARIANCE CURVES

After using the caudal characteristic curve to determine which packet length/detection time ratio is needed, the next step is to decide what throughput is desired and what value of expected packet delay can be tolerated. As in most engineering problems, a tradeoff is involved. The constant expected delay and variance curves we present here will help to decide what tradeoff will suit the designer's application.

In Figure 3, maximum throughput curves for various expected delay values are shown. An operating point under a given curve will have an expected delay that is less than the given bound.

The variance of delay for a finite population in a CSMA/CD network can be column corresponding to the absorbing state z found using a result in [10]. It is given by

$$\text{var}(D) = 2\alpha \mathbf{T}^{-2}\mathbf{e} - E^2[D] \tag{4.1}$$

where

$$E[D] = -\alpha \mathbf{T}^{-1}\mathbf{e} \tag{4.2}$$

and where α is the row vector containing the initial probability distribution, $\mathbf{T}$ is the generator matrix $\mathbf{A}$ with the row and deleted, and $\mathbf{e}$ is a column vector containing all ones. Figure 4 shows one example of the variance vs. load curve. The curve is relatively flat for loads that would yield the higher throughputs. Hence, the variance of delay does not fluctuate much for loads around the operating point. For the purpose of comparison, the variance vs. load curve of a network similar to our model, but with a first come, first served (FCFS) service order, is given in Figure 5. Note that the variance of the CSMA/CD network for a given load, is greater than the variance of the FCFS model. This result is expected, since a packet's delay time is affected by how often it collides with another packet.

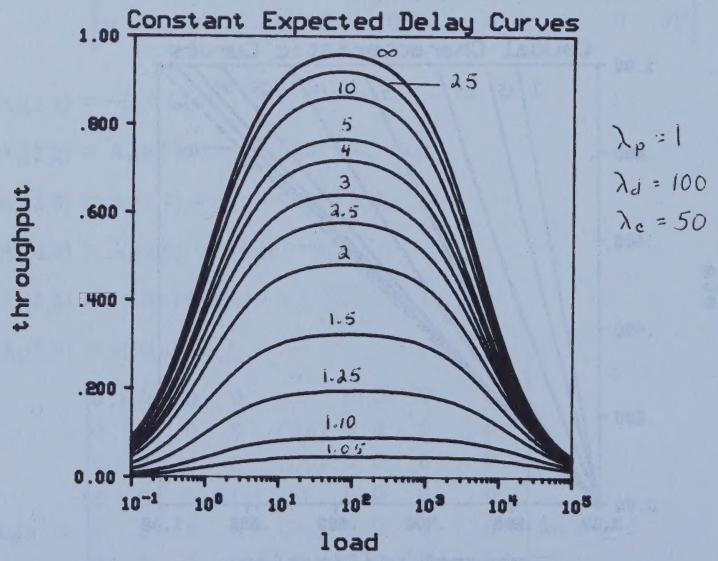


Figure 3: Throughput vs. Channel loading for the CSMA/CD protocol for various expected delays. The (packet length)/(detection time) ratio is 100.

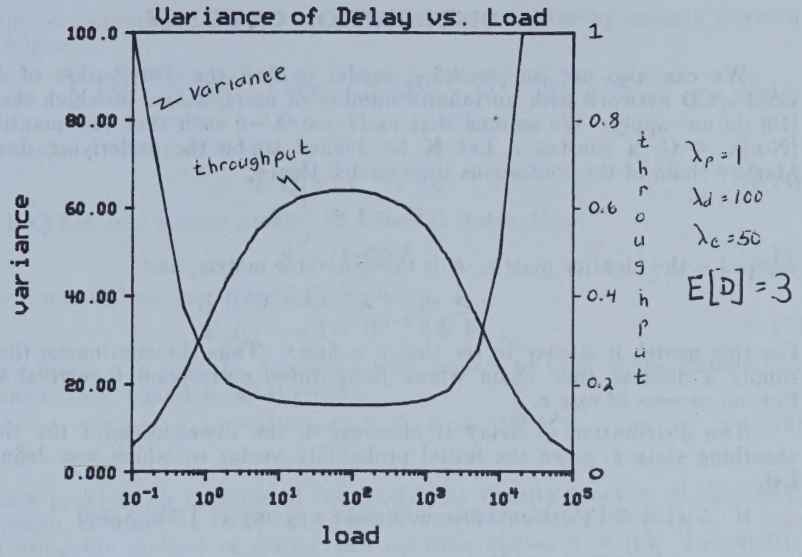


Figure 4: Variance of delay and throughput vs. channel loading for the CSMA/CD protocol with the expected delay held constant. The (packet length)/(detection time) ratio is 100, and the expected delay is 3 packet lengths.

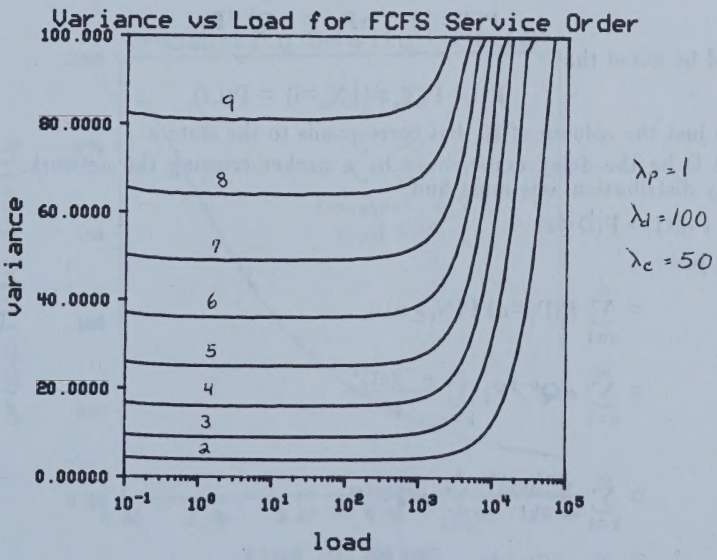


Figure 5: Variance of delay vs. channel loading for network with a first come first served service order for various expected delays.

V. DISTRIBUTION OF DELAY

We can also use our modified model to find the distribution of delay in a CSMA/CD network with an infinite number of users, a case in which the results in [10] do not apply. We assume that as $N\to\infty$, $\lambda\to 0$ such that the quantity $G(n) = (N-n)\lambda = G$, a constant. Let $\mathbf{K}$ be defined to be the underlying discrete time Markov chain of the continuous time model. Hence,

$$\mathbf{K} = \mathbf{I} + \frac{1}{c}\mathbf{A} \tag{5.1}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{A}$ is the generator matrix, and

$$c \geq \sup_{i \geq 0} \{ \lambda_i : \lambda_i = -A(i,i) \}. \tag{5.2}$$

For this model, it is easy to see that c is finite. Thus the continuous time chain is simply a discrete time chain whose jump times correspond to arrival times of a Poisson process of rate c.

The distribution of delay is identical to the distribution of the time to the absorbing state z, given the initial probability vector $\boldsymbol{\alpha}$, which was defined earlier. Let.

$$F_k(i,z) = P(\text{first entrance into state z occurs at } T=k \mid X_0=i) \tag{5.3}$$

$$= P(T_z=k \mid X_0=i) \tag{5.4}$$

and let $\mathbf{F}_k$ be a column vector where $\mathbf{F}_k(i) = F_k(i,z)$. Then $\boldsymbol{\alpha}\mathbf{F}_k = P(T_z=k)$. It has been shown [11] that

$$\mathbf{F}_{k+1} = \mathbf{Q}\mathbf{F}_k \tag{5.5}$$

where $\mathbf{Q}$ is the matrix $\mathbf{K}$ with the column corresponding to state z set to zeroes. Hence

$$P(T_j=k) = \boldsymbol{\alpha}\mathbf{F}_k = \boldsymbol{\alpha}\mathbf{Q}^{k-1}\mathbf{F}_1. \tag{5.6}$$

It should be noted that

$$\mathbf{F}_1 = P(T_z=1 \mid X_0=i) = P(i,z), \tag{5.7}$$

which is just the column of $\mathbf{K}$ that corresponds to the state z.

Let D be the delay experienced by a packet crossing the network. To obtain the delay distribution, one must find

$$F_D(t) = P(D \leq t) \tag{5.8}$$

which is just

$$= \sum_{n=1}^{\infty} P(D_j=n)\, P(N_t \geq n) \tag{5.9}$$

$$= \sum_{n=1}^{\infty} \boldsymbol{\alpha}\mathbf{Q}^{n-1}\mathbf{F}_1 \sum_{k=n}^{\infty} \frac{e^{-ct}(ct)^k}{k!} \tag{5.10}$$

$$= \sum_{k=1}^{\infty} \frac{e^{-ct}(ct)^k}{k!} \sum_{n=1}^{k} \boldsymbol{\alpha}\mathbf{Q}^{n-1}\mathbf{F}_1 \tag{5.11}$$

$$= \sum_{k=1}^{\infty} \frac{e^{-ct}(ct)^k}{k!}\, \boldsymbol{\alpha} \left(\sum_{n=0}^{k-1} \mathbf{Q}^n \right) \mathbf{F}_1. \tag{5.12}$$

The quantity in parenthesis in equation (5.12) can be rewritten by using a theorem in [11], as follows

$$\sum_{n=0}^{k-1} \mathbf{Q}^n = \sum_{n=0}^{\infty} \mathbf{Q}^n - \sum_{n=k}^{\infty} \mathbf{Q}^n = [\mathbf{I}-\mathbf{Q}^k] \sum_{n=0}^{\infty} \mathbf{Q}^n = [\mathbf{I}-\mathbf{Q}^k]\,\mathbf{S} \tag{5.13}$$

where $\mathbf{S}$ is the minimal solution of

$$(\mathbf{I}-\mathbf{Q})\mathbf{Y}=\mathbf{I} \quad \mathbf{Y}\geq 0. \tag{5.14}$$

Note that if $\mathbf{Q}$ has only a finite number of transient states, then

$$\mathbf{S} = (\mathbf{I}-\mathbf{Q})^{-1}. \tag{5.15}$$

After some computations, equation (5.12) becomes

$$F_D(t) = \alpha[\mathbf{I}-e^{-ct(\mathbf{I}-\mathbf{Q})}]\,\mathbf{S}\,\mathbf{F}_1. \tag{5.16}$$

The density function is just the derivative with respect to time of the distribution function which, from equation (5.14) yields

$$f_D(t) = c\alpha e^{-ct(\mathbf{I}-\mathbf{Q})}(\mathbf{I}-\mathbf{Q})\,\mathbf{S}\,\mathbf{F}_1 = c\alpha e^{-ct(\mathbf{I}-\mathbf{Q})}\mathbf{F}_1. \tag{5.17}$$

Figure 6 provides an example of the probability density function of the delay, given a specific throughput, load, and packet length/detection time ratio. It was calculated using the method of scaling and squaring discussed in [13]. The matrix $-ct(\mathbf{I}-\mathbf{Q})$ was computed for a specific value of t. Then it was divided by a power of two, say 2^n, until all of its eigenvalues had a magnitude less than one. The exponential of the matrix was then computed using a Taylor series, only seven terms of which were needed to ensure high accuracy. Lastly, the result was taken to the 2^n power. From there, the density function at the value t was computed. This technique was repeated until enough points were computed to obtain Figure 6.

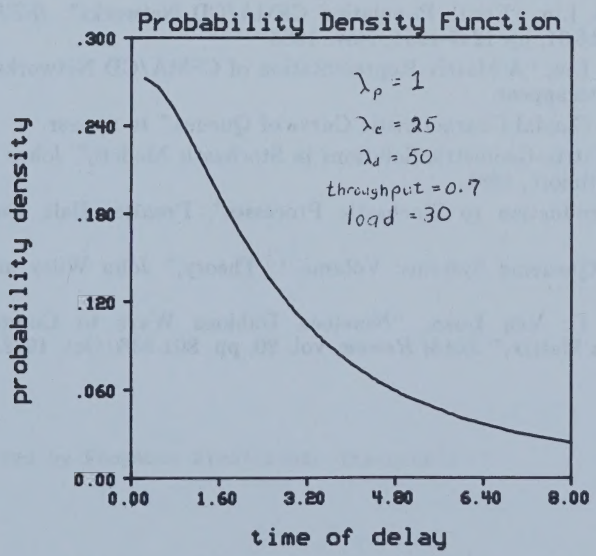


Figure 6: Probability density function of the delay of a packet on the CSMA/CD network.

VI. CONCLUSION

This paper develops new tools for the design of CSMA/CD networks; the caudal characteristic curve, delay variance and distribution, and constant expected delay curves. The caudal characteristic curve can be used to decide which packet length/detection time ratio is adequate. The constant expected delay and variance curves, show the tradeoffs between throughput and packet delays. They also show which load will yield the highest throughput for a given expected delay. Finally, the distribution of delay will show exactly what type of behavior should be expected from a given network.

VII. REFERENCES

[1] R. Metcalf and D. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks", *CACM*, pp 395-404, July 1976.

[2] C. Bass, J. Kennedy, and J. Davidson, "Local Network Gives New Flexibility to Distributed Processing", *Electronics*, pp 114-122, Sept. 25, 1980.

[3] L. Kleinrock and F. Tobagi, "Packet Switching in Radio Channels: Part I, Carrier Sense Multiple Access Modes and Their Throughput Delay Characteristics", *IEEE Trans. Commun.*, vol COM-23, pp 1400-1416, Dec. 1975.

[4] F. Tobagi and V. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection", *Computer Networks*, vol 4, pp 245-259, Oct/Nov. 1980.

[5] J.S. Meditch and C.T.A. Lea, "Stability and Optimization of the CSMA and CSMA/CD Channels," *IEEE Trans. Commun.*, vol COM-31, pp 763-774, June 1983.

[6] I. Rubin, "Synchronous and Carrier Sense Asynchronous Dynamic Group Random Access Schemes For Multiple Access Communications," *IEEE Trans. Comm.*, vol COM-31, pp 1063-1077, Sept. 1983.

[7] E. Coyle and B. Liu, "Finite Population CSMA/CD Networks", *IEEE Trans. Comm.*, vol COM-31, pp 1247-1251, Nov. 1983.

[8] E. Coyle and B. Liu, "A Matrix Representation of CSMA/CD Networks," *IEEE Trans. Comm.*, to appear.

[9] M. Neuts, "The Caudal Characteristic Curve of Queues," to appear.

[10] M.F. Neuts, "Matrix-Geometric Solutions in Stochastic Models," Johns Hopkins Univ. Press, Baltimore, 1981.

[11] E. Cinlar, "Introduction to Stochastic Processes", Prentice Hall, Englewood Cliffs, NJ, 1975.

[12] L. Kleinrock, "Queueing Systems: Volume 1: Theory," John Wiley and Sons, New York, 1975.

[13] C. Moler and C. Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix," *SIAM Review*, vol. 20, pp. 801-836, Oct. 1978.

TWO-COMMODITY NETWORK DESIGN WITH MULTIPLE DEMAND

Mario LUCERTINI - Giuseppe PALETTA

IASI-CNR e
Università di
Roma "La Sapienza"
Roma, Italy

CRAI
Via Bernini, 5
Rende (CS)
Italy

ABSTRACT

This paper presents and solves the problem of finding the minimum total cost edge capacities, such that a network satisfies non-simultaneously a set of two-commodity demand vectors.

A polynomial algorithm is given based on a decomposition of the optimal network as sum of simple paths between the network endpoints.

Supported by Progetto Finalizzato Trasporti.

1. INTRODUCTION

Network Design problems are among the most important problems in the design of distributed computing systems and communication systems (1,2).

Most of the authors formulate the problem as that of finding edge capacities such that time varying demands are satisfied by the flow network (3,4). In (5,6,7) the problem of constructing a network to meet many flow demands non-simultaneously on a set of sink nodes, is considered for single commodity. The problem can be easily formulated as a linear programming problem but the standard solution procedures based on simplex method are quite heavy and lead in many pratical cases to degenerate problems.

In (7) a different approach based on a representation of the optimal network as sum of so called basic networks (paths, trees or cycles), both for directed and non directed networks, is proposed and some results on the properties of the optimal solution are given.

The network design problem with multiple demands, to be satisfied non-simultaneously, occurs in all contexts where the network must meet a given number of demands occurring casually along the time horizon under consideration.

In this paper a polynomial algorithm to solve two-commodity network design problems with multiple demands to be satisfied non-simultaneously is developed. Only non directed networks are considered, but all results can be extended to the directed case.

2. PROBLEM AND OUTLINE OF PREVIOUS RESULTS

Let G(N,A,c) be a non directed network where N is the set of n nodes, A is the set of m arcs, and c is an m-vector representing the capacity of the arcs.

A commodity k is characterized by a node (the source) s_k and a node (the sink) p_k. Suppose that 2 commodities flow through the network. The total flow through an arc (equal to the sum of the flows of all commodities through this arc) is bounded above by the capacity of the arc.

Given an m-vector $g=(g_1,..,g_m)$, indicating the unitary capacity expansion costs and a set $D=\{d^1,d^2,..,d^r\}$ of two-commodity demands vectors, in which

d_1^i represents the i-th demand of commodity 1 and d_2^i the i-th demand of commodity 2. The problem is: find the minimum cost edge capacities such that the network G(N,A,c) satisfies non-simultaneously all demand vectors d^i in a given set D.

In (4) for two-commodity flow problem is proved the following theorem that results useful to formulate the considered problem as a easy linear programming problem.

Let T_1 be the minimum capacity of a cut set separating the nodes s_1 and p_1, T_2 be the same for the nodes s_2 and p_2, and T_{12} be the minimum capacity separating both the nodes s_1 and p_1, and the nodes s_2 and p_2. Note that $T_1+T_2 \geq T_{12}$. Then we have:

THEOREM (Hu). A two-commodity flow of amount d_1 in the first commodity and d_2 in the second commodity is attainable if and only if

$$d_1 \leq T_1$$
$$d_2 \leq T_2$$
$$d_1+d_2 \leq T_{12}$$

This theorem enables us to obtain the following Lemma.

If

$$dm1 = \max_i \; d_1^i$$
$$dm2 = \max_i \; d_2^i$$
$$dm12 = \max_i \; d_1^i + d_2^i$$

then we have:

LEMMA. The demand vectors $d \in D$ are non-simultaneously satisfied if and only if $T_1 \geq dm1$

$T_2 \geq dm2$

$T_{12} \geq dm12$

Proof. If the inequalities are satisfied then all the demand vectors (theorem 1) can be satisfied. On the other hand, if all the demand vectors can be satisfied the capacities of the minimal cuts are at least dm1, dm2 and dm12.

The problem can be expressed as follows:

Find the minimum total cost edge capacity such that the constraints of

the Lemma will be satisfied.

3. THE ALGORITHM

For each couple (s_j, p_j), j=1,..,2 let us consider the set of all elementary chains between s_j and p_j: $P_1^j,..,P_{z(j)}^j$ (where z(j) is the total number of such chains). Let C_i^j denote the capacity on each arc belonging to P_i^j chain.

Also, let us consider the set of all cycles connecting the nodes s_1, p_1, s_2 and p_2: $P_1^{12},..,P_{z(12)}^{12}$ (where z(12) is the total number of such chains). Let C_i^{12} denote the capacity on each arc belonging to P_i^{12} cycle.

It is easy to verify that:

$$T_1 = \sum_{j=1}^{z(1)} C_j^1 + 2 \sum_{j=1}^{z(12)} C_j^{12}$$

$$T_2 = \sum_{j=1}^{z(2)} C_j^2 + 2 \sum_{j=1}^{z(12)} C_j^{12}$$

$$T_{12} = \sum_{j=1}^{z(1)} C_j^1 + \sum_{j=1}^{z(2)} C_j^2 + 2 \sum_{j=1}^{z(12)} C_j^{12}$$

Also, if L_j^1, L_j^2 and L_j^{12} are respectively the length of the chains P_j^1, P_j^2 and of the cycle P_j^{12}, then the total cost of the capacity is equal to

$$\sum_{j=1}^{z(1)} L_j^1 C_j^1 + \sum_{j=1}^{z(2)} L_j^2 C_j^2 + \sum_{j=1}^{z(12)} L_j^{12} C_j^{12}.$$

The problem can be written as a linear programming problem:

$$\min \sum_{j=1}^{z(1)} L_j^1 C_j^1 + \sum_{j=1}^{z(2)} L_j^2 C_j^2 + \sum_{j=1}^{z(12)} L_j^{12} C_j^{12}$$

$$\sum_{j=1}^{z(1)} C_j^1 + 2 \sum_{j=1}^{z(12)} C_j^{12} \geq dm1$$

$$\sum_{j=1}^{z(2)} C_j^2 + 2 \sum_{j=1}^{z(12)} C_j^{12} \geq dm2$$

$$\sum_{j=1}^{z(1)} C_j^1 + \sum_{j=1}^{z(2)} C_j^2 + 2 \sum_{j=1}^{z(12)} C_j^{12} \geq dm12$$

$C_j^1 \geq 0 \qquad j=1,..,z(1)$

$C_j^2 \geq 0 \qquad j=1,..,z(2)$

$C_j^{12} \geq 0 \qquad j=1,..,z(12)$

The optimal solution is the following:

If $2(L_1+L_2)>L_{12}$

$$C_j^1 = \begin{cases} dm12-dm2 \text{ if } P_j^1 \text{ is the shortest path between } s_1 \text{ and } p_1 \\ 0 \quad \text{otherwise} \end{cases}$$

$$C_j^2 = \begin{cases} dm12-dm1 \text{ if } P_j^2 \text{ is the shortest path between } s_2 \text{ and } p_2 \\ 0 \text{ otherwise} \end{cases}$$

$$C_j^{12} = \begin{cases} (dm1+dm2-d\ 12)/2 \text{ if } P_j^{12} \text{ is the shortest cycle} \\ 0 \text{ otherwise} \end{cases}$$

If $2(L_1+L_2)< L_{12}$

$$C_j^1 = \begin{cases} dm1 \text{ if } P_j^1 \text{ is the shortest path between } s_1 \text{ and } p_1 \\ 0 \text{ otherwise} \end{cases}$$

$$C_j^2 = \begin{cases} dm2 \text{ if } P_j^2 \text{ is the shortest path between } s_2 \text{ and } p_2 \\ 0 \text{ otherwise} \end{cases}$$

$C_j^{12} = 0$

Where

L_1 is the length of the shortest path between s_1 and p_1

L_2 is the length of the shortest path between s_2 and p_2

L_{12} is the length of the shortest cycle

The optimal solution of the former linear programming problem leads to the following theorem:

THEOREM. The optimal solution of the two-commodities network design problem can be obtained as the superposition of three basic networks (with uniform capacity):

i) the shortest path from s_1 to p_1;

ii) the shortest path from s_2 to p_2;

iii) the minimum lenght cycle among s_1, p_1, s_2, p_2.

The algorithm is the following, where P_1, P_2, and P_{12} are, respectively, the shortest path between s_1 and p_1, between s_2 and p_2 and the shortest cycle.

ALGORITM A

STEP 0. Find dm1, dm2, dm12

Find P_1, P_2, P_{12}

(L_1, L_2, L_{12})

STEP 1. If $2(L_1 + L_2) \leq L_{12}$

then set

$$C^1_{ij} = dm1 \qquad \forall\ (i,j) \in P_1$$

$$C^2_{ij} = dm2 \qquad \forall\ (i,j) \in P_2$$

else set

$$C^1_{ij} = dm12 - dm2 \quad \forall\ (i,j) \in P_1$$

$$C^2_{ij} = dm12 - dm1 \quad \forall\ (i,j) \in P_2$$

$$C^{12}_{ij} = (dm1 + dm2 - dm12)/2 \quad \forall\ (i,j) \in P_{12}$$

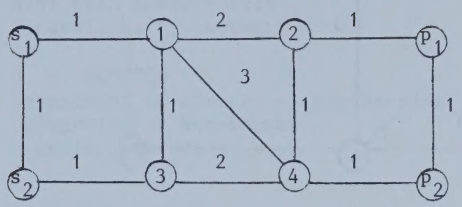
STEP 2. Set $C_{ij} = C^1_{ij} + C^2_{ij} + C^{12}_{ij} \quad \forall\ (i,j) \in A$

Stop.

As far as the computational complexity is concerned, to compute $(P_1, P_2, P_{12}, L_1, L_2, L_{12})$ requires $O(n^2)$ time, to compute dm1, dm2, dm12 requires $O(r)$ time; step 1 and step 2 require $O(n)$ time.

Example

Given the following network



where the number beside each arc indicates its capacity expansion cost.

The demand vectors, to satisfy non-simultaneously, are the following:

$$d = \binom{4}{1}, \quad d^2 = \binom{1}{4}, \quad d^3 = \binom{3}{3}.$$

In this case we obtain :

dm1= 4

dm2= 4

dm12=6

Also we obtain:

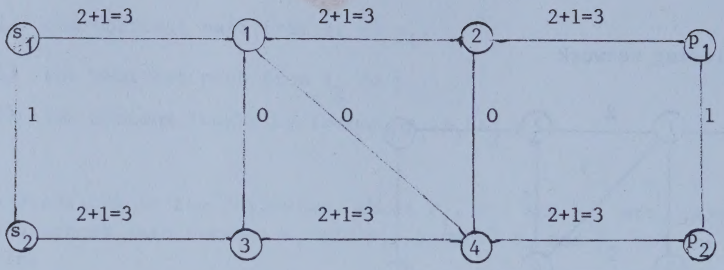
$P_1=(s_1,1,2,p_1)$, $L_1=4$

$P_2=(s_2,3,4,p_2)$, $L_2=4$

$P_{12}=(s_1,1,2,p_1,p_2,4,3,s_2)$, $L_{12}=10$

Because $2(L_1+L_2) > L_{12}$, we provide 2 units of capacity in each arc belonging to P_1; 2 units of capacity in each arc belonging to P_2; and 1 unit of capacity in each arc belonging to P_{12}.

The network that at minimum cost meets non-simultaneously the demand vectors is the following:



where the number beside each arc indicates its capacity. The minimum cost is equal to 26.

REFERENCES

(1) M. Kobayashy, Communication network design and control algorithms: a survey, Nato Advanced Study Institute, Norwich, 1980.

(2) J. Freidenfelds, Capacity expansion, North Holland, 1981.

(3) R.E. Gomory, T.C. Hu, Synthesis of a communication network, J. of S.I.A.M., 12, 348-369 (1964).

(4) T.C. Hu, Integer programming and network flows, Addison Wesley, 1970.

(5) M. Lucertini, G. Paletta, A class of network design problems with multiple demand: Model formulation and an algorithmic approach, Mathematical Programming Study (to appear).

(6) M. Lucertini, G. Paletta, Network design with multiple demands: a new approach, Annals of discrete Mathematics, North Holland (to appear).

(7) M. Lucertini, G. Paletta, Network design with non-simultaneous flows, In Algorithm design for computer system design, Springer Verlag, 1984.

(8) E. Lawler, Combinatorial optimization networks and matroids, Holt Rinehart and Winston, 1976.

A PERFORMANCE COMPARISON OF THE TOKEN RING, TOKEN BUS, AND CSMA/CD BUS*

C. T. A. LEA
AT&T Bell Laboratories
Naperville, IL 60566

J. S. MEDITCH
Department of Electrical Engineering
University of Washington
Seattle, WA 98195

ABSTRACT

The token ring, token bus, and CSMA/CD bus have emerged as three systems possessing major interest for applications in local area networking. In this paper, a comparison of the delay vs. throughput performance is presented for these three systems under a common set of modeling assumptions, traffic loads, and channel bandwidths. The approach is that of determining lower bounds on performance and identifying associated tradeoffs among the systems.

1. INTRODUCTION

Due to their simplicity, together with a number of related design and implementation considerations, the ring and bus are two of the most popular topologies for local area networks [1-4]. One of the key issues in the design and implementation of such networks is that of providing a reliable and inexpensive controller for efficient sharing of the communications channel by the stations. A number of methods addressing this controller/channel access protocol problem have been proposed, each offering a different tradeoff between efficiency and complexity (cost). The result has been the emergence of the token ring, token bus, and CSMA/CD bus as three systems possessing major applications and interest in the industry [5,6].

In this paper, we present a comparison of the delay vs. throughput performance for these three systems under a common set of modeling assumptions, traffic loads, and channel bandwidths. The approach is that of determining lower bounds on performance and identifying associated tradeoffs among the systems.

The paper is organized as follows. We begin in Section 2 with a description of the mathematical models for the token ring, token bus, and CSMA/CD bus. Numerical results using these models are then presented and discussed in Section 3, and we conclude in Section 4 with a summary of our work and some remarks on related aspects of local area network implementation.

2. NETWORK MODELS

Token Ring

We consider a token ring network with queueing in which packets arrive at each of N equally spaced stations in a Poisson stream with average rate

*Research sponsored in part by the National Science Foundation under NSF Grant ECS-8011262.

λ/N packets per sec. Upon receipt of the token, a station with one or more packets in its queue will remove the token from the ring, transmit one packet, await its return, and then pass the token to its adjacent downstream neighbor. If its queue is empty, a station forwards the token to its neighbor subject to the delay described below. Packet length y is in bits and may be either fixed or an independent, identically distributed random variable with mean $\overline{y}$ and second moment $\overline{y^2}$. It is assumed that each packet may also include a header of fixed length h. This gives a total packet length of y + h bits, and a service time of x = (y+h)/C where C is the channel capacity in bits/sec. The time required to pass the token from one station to its adjacent downstream neighbor is assumed to consist of the link's propagation delay plus the latency time in the first station. This is denoted by δ in sec, the resulting total delay around the ring then being $\tau = N\delta$ sec. Under these conditions, the average packet delay in sec is [7,8]

$$T = \overline{x} + \frac{\lambda \overline{x^2}}{2(1-\rho)} + \frac{\tau(1-\frac{\rho}{N})}{2(1-\rho)} \qquad (1)$$

where $\rho = \lambda\overline{x}$ is the average utilization.

Token Bus

In this type of network, N stations are attached to a passive bus. Access to the channel is granted to Station i + 1 by its predecessor, Station i, via the latter's transmission of a token frame which must contain Station i + 1's address as well as the token itself since we are using a broadcast channel. In general, a station holding the token may transmit packets for a specified maximum "token-hold" time. When this time period expires or the station has no more packets to transmit, it passes the token to the next station in the sequence. A virtual ring is formed when Station N passes the token to Station 1. We note that the token frame is itself a packet which is handled separately from data packets.

A simplified delay analysis which omits the effects of queueing delay is given in [4] where it is assumed that a station may hold the token for up to a limited time of Δ sec. The maximum time for a token to make a complete cycle of the stations is readily seen to be

$$C = N\Delta + \sum_{j=1}^{N} \tau_j$$

where τ_j in sec is the switchover time between successive stations. The maximum time between successive accesses by any Station i, i = 1, • • •, N, is then obviously given by

$$A_i = (N-1)\Delta + \sum_{j=1}^{N} \tau_j$$

We observe that a queueing system analysis is possible here if we replace the assumption on Δ by one in which a station holds the token only for the duration of a single packet transmission, if any. This parallels the token passing protocol for the ring network above, and the result is a virtual token ring with average packet delay

$$T = \overline{x} + \frac{\lambda \overline{x^2}}{2(1-\rho)} + \frac{N\sigma(1-\frac{\rho}{N})}{2(1-\rho)} \qquad (2)$$

in sec where λ is the same as in Eq. (1), $x = y/C + d$, $\rho = \lambda\overline{x}$, d is the one-way propagation delay on the bus in sec, and σ is the station-to-station switchover time which is described below.

For N equally spaced stations on the bus, it is easy to see that the average propagation delay in token passing is minimized if the stations are addressed 1, N, 2, N-1, • • •, N/2, N/2+1 from left to right on the bus and the token passing sequence is $1 \rightarrow 2 \rightarrow 3 \rightarrow \cdots \rightarrow N \rightarrow 1$ where we assume that N is even. The resulting average propagation delay between adjacent stations is 2d/N. To this we add the token transmission time f/C and the token latency γ to obtain

$$\sigma = \frac{2d}{N} + \left(\frac{f}{C} + \gamma\right)$$

In this expression, f is the token frame length in bits, C the channel capacity in bits/sec, and γ is in sec. Since we shall use a station latency of 1 bit in both the token ring and token bus, we have $\gamma = 1/C$.

CSMA/CD Bus

For this system, we consider the slotted, nonpersistent, CSMA/CD channel which was studied in detail in [9]. We summarize only the basic modeling assumptions and the equations which must be evaluated to determine the average packet delay.

The time axis is partitioned into minislots of length d sec, the one-way propagation on the bus. Time is expressed in minislots with packet length taken to be y minislots and the collision detection (CD) time fixed at $R < y$ minislots. A large number of stations are assumed to collectively generate a Poisson stream of packets with average rate λ packets/minislot.

Under the nonpersistent CSMA protocol, stations with a packet ready to transmit proceed as follows. If the channel is sensed idle, a station transmits its packet at the beginning of the next minislot. On the other hand, if a station finds the channel busy, it reschedules its sensing of the channel for some future minislot according to a specified retransmission policy. When two or more stations sense the channel idle in the same minislot, their ensuing transmissions will interfere with each other and result in a collision. Stations which have been involved in a collision or found it necessary to defer transmission upon sensing the channel busy are said to be blocked. In the absence of a suitable retransmission policy which the blocked stations follow, this channel is unstable in the sense of an eventual deadlock with all stations in the blocked state. In [9], it was shown that the random retransmission policy $p = \alpha/n$ where $n = 1, 2, \cdots$ is the number of blocked stations could stabilize the channel as a function of the other channel parameters. The range of values on $\alpha \in (0,1]$ sufficient to do so is determined from a stability inequality with α then chosen to optimize delay vs. throughput performance.

The delay-throughput relation for the stabilized nonpersistent CSMA/CD channel was developed in [9] using an imbedded Markov chain in which the imbedded times occur at the end of successful transmissions. The chain's state was taken to be the number of blocked stations left behind by a successful transmission, i.e., the size of the channel queue at the

imbedded times. Once the average queue size $\overline{q}$ is known, the average packet delay is determined via Little's result [10].

The evaluation of $\overline{q}$ proceeds as follows. Letting Q(z) denote the moment generating function for the number of packets in the channel, it can be shown [9] that

$$Q(z) = \frac{\{Q_o[U_o(z)-U(z)]+Q_1 z[U_1(z)-U(z)]\}V(z)}{z - U(z)\,V(z)}$$

In this equation, $U(z) = Y(\lambda-\lambda z)$, $U_1(z) = Y_1(\lambda-\lambda z)$, $V(z) = \exp[-(\lambda-\lambda z)\overline{y}]$, and

$$U_o(z) = \frac{c_1 z}{1-c_o} + \frac{1}{1-c_o}\left[\left(e^{-(\lambda-\lambda z)} - c_o - c_1 z\right) \cdot Y(\lambda-\lambda z)e^{(R+1)(\lambda-\lambda z)}\right]$$

where

$$Y(s) = \frac{(1-P_o)e^{-s}}{1-P_o e^{-s}}\, J\left[\frac{(1-P_o)e^{-(R+2)s}}{1-P_o e^{-s}}\right],$$

$$J(z) = \frac{P_1}{1-(1-P_1)z}, \qquad Y_1(s) = \frac{\alpha e^{-s}}{1-(1-\alpha)e^{-s}}, \qquad P_o = e^{-G},$$

$$P_1 = \frac{Ge^{-G}}{1-e^{-G}}, \qquad c_o = e^{-\lambda}, \qquad c_1 = \lambda e^{-\lambda},$$

$\overline{y}$ is the average packet length in bits, and $G = \lambda + \alpha$ is the total offered traffic (new plus rescheduled packets, respectively) in packets/minislot.

The values of Q_o and Q_1 are determined from a pair of linearly independent algebraic equations which follow from the facts that $Q(1) = 1$ and $dQ(z)/dz = Q_1$ when $z = 0$. From the moment generating properties of Q(z), the average number of packets in the channel queue is $\overline{q} = dQ(z)/dz$ evaluated at $z = 1$ and we have that the average packet delay in sec is

$$T = \frac{\overline{q}d}{\lambda} \tag{3}$$

3. NUMERICAL RESULTS

Delay vs. throughput performance results obtained using the above models are presented in Figs. 1 through 3. In these figures, the ordinate gives average delay T relative to average packet transmission time $\overline{y}/C$, while the scale for the abscissa is that of average throughput rate $\lambda\overline{y}$ relative to channel capacity C. We thus show normalized average delay $D \triangleq CT/\overline{y}$ vs. normalized average throughput $S \triangleq \lambda\overline{y}/C$. This scaling is a matter of convenience and should be kept in mind when comparing performance.

For all three systems, we assume a 2 km coaxial cable with a propagation delay of 5 μsec/km and a total average arrival rate of λ packets/sec.

For the token ring and token bus, we take N = 200 stations and a 1 bit latency. The other parameter values that we use are channel capacities of 1, 5, and 10 Mbits/sec (MBPS); packet length either constant at y = 1000 bits or exponentially distributed with $\bar{y}$ = 1000 bits; a token frame length of f = 12 or 24 bits for the token bus; and a collision detect time of R = 4 minislots for the CSMA/CD bus. Further, we take the packet header to be h = 0 thereby making the packet length identical for all three systems. The specific values of all parameters are noted on each figure and/or in its caption, and are mentioned in the text where the results are discussed.

In Fig. 1, we show the performance of the token ring, the token bus with 12 and 24 bit token frames, and the CSMA/CD bus with a CD time of 4 minislots. The channel capacity is 1 MBPS and the packet length is fixed at 1000 bits. Under the assumptions here, the token ring and the CSMA/CD bus have virtually the same performance up to about S = 0.7. Beyond that, the ring performs better, especially for S > 0.85. The delay-throughput performance of the token bus relative to CSMA/CD is competitive only for S in the vicinity of 0.90, and not so at all compared with the ring over the entire range on S. The basic difficulty with the token bus is the fixed extra delay in token passing since the token frame is a separate packet which carries the token itself plus the address of the next station which is to have access to the channel. This causes the switchover time to be higher than in the token ring. The effect on delay-throughput behavior for the token bus as a function of token frame length is clear from the figure as we move from h = 24 bits to 12 bits.

Figure 2 shows the performance characteristics for the same networks as in Fig. 1 except for the channel capacity which is now 10 MBPS. The advantage of the ring over the token bus is clear; that of the ring over the CSMA/CD bus must be qualified. In both Figs. 1 and 2, the value of R is 4 minislots. For the channel capacity C = 1 MBPS used in generating the data for Fig. 1, this translates to 40 bits; for C = 10 MBPS, relevant to Fig. 2, this is 400 bits. In the first case, the CD time is 4% of the packet length; in the second, it is 40%. It is known [9] that CD packet length relative to data packet length has a significant effect on delay-throughput behavior in CSMA/CD. This is amply demonstrated here. Nominally, we can expect the characteristics of the token ring and the CSMA/CD bus to compare as in Fig. 1 if the ratio of CD packet length to data packet length is kept constant while bandwidth is changed.

Finally, we show in Fig. 3 the delay-throughput behavior of the networks for two different types of packets. Specifically, we assume packets of fixed length 1000 bits as before, and those which are exponentially distributed in length with mean 1000 bits. For the token bus, we let f = 24 bits, and for the CSMA/CD bus, we again take R = 4 minislots. The channel capacity is 5 MBPS. The effect of variable packet length relative to fixed packet length is clear from the figure. The result is more pronounced for the ring as a function of increasing throughput than it is for the others, but the overall performance comparison remains essentially as before.

4. CONCLUSION

Under the modeling assumptions and channel parameter values used here, the performance advantage resides with the token ring. The CSMA/CD bus is competitive for low to moderate throughput, 0 < S < 0.7, for sufficiently short CD times (see Fig. 1). The performance of the token bus is hampered by the need for the token frame to clear the channel before the station for

which the token is intended can transmit its data packet. We observe that token passing on the bus is particularly "expensive" at low throughput values. On the other hand, relative to the CSMA/CD bus, the token bus has better delay-throughput behavior at higher throughput with R fixed as is clear for the cases in Figs. 2 and 3. This is to be expected since token passing on the bus is a small fraction of the traffic at higher throughput and dedicated channel access is provided to the stations, whereas collision resolution is a major factor for the CSMA/CD bus for large S. Finally, with respect to these conclusions, it should be borne in mind that all parameters for the three systems are fixed except for two values of f, three values of C, and the range on the packet arrival rate λ. Hence, it does not follow that the above comparison applies for shorter or longer packets, e.g., 200 to 500 and 2000 to 5000 bits, respectively, different cable lengths, or other values of R.

Our study has focused on performance characteristics at the link level under error-free transmission. Hence, among other things, delays due to error control, system initialization, and recovery from system failures are not included. While these are handled at the higher protocol levels, we note that the token ring and token bus are subject to failure as the result of a "garbled" or "lost" token, but there is no such problem with the CSMA/CD bus. In a similar way, a station failure on the CSMA/CD bus will not cause a network failure whereas it can in the other two systems unless a reliable "bypass" and recovery mechanism is provided. Other implementation factors include hardware and software development, ease of expansion or contraction, maintenance, cost, and associated tradeoffs. The present study then, just as others like it, serves to provide a basic understanding of the performance limits of candidate systems.

REFERENCES

1. D. D. Clark, K. T. Pogran, and D. P. Reed, "An introduction to local area networks," Proc. IEEE, vol. 66, pp. 1497-1517, Nov. 1978.

2. D. R. Boggs and R. M. Metcalfe, "Ethernet: Distributed packet switching for local computer networks," Comm. Ass. Comput. Mach., vol. 19, pp. 395-404, July 1976.

3. J. H. Saltzer and D. D. Clark, "Why a ring?," Proc. 7th Data Comm. Symp., Mexico City, Oct. 1981, pp. 211-217.

4. C. K. Miller and D. M. Thompson, "Making a case for token passing in local networks," Data Commun., vol. 11, pp. 79-88, Mar. 1982.

5. G. Clare, "The IEEE 802 committee states its case concerning its local network standards efforts," Data Commun., vol. 11, p. 13, Apr. 1982.

6. M. Graube, "Local area nets: a pair of standards," IEEE Spectrum, vol. 19, pp. 60-64, June 1982.

7. A. G. Konheim and B. Meister, "Waiting lines and times in a system with polling," J. Ass. Comput. Mach., vol. 21, pp. 470-490, July 1974.

8. W. Bux, "Local-area subnetworks: a performance comparison," IEEE Trans. Commun., vol. COM-29, pp. 1465-1473, Oct. 1981.

9. J. S. Meditch and C. T. A. Lea, "Stability and optimization of the CSMA and CSMA/CD channels," IEEE Trans. Commun., vol. COM-31, no. 6,

pp. 763-774, June 1983.

10. L. Kleinrock, Queueing Systems, Vol. I,: Theory. New York: Wiley-Interscience, 1975.

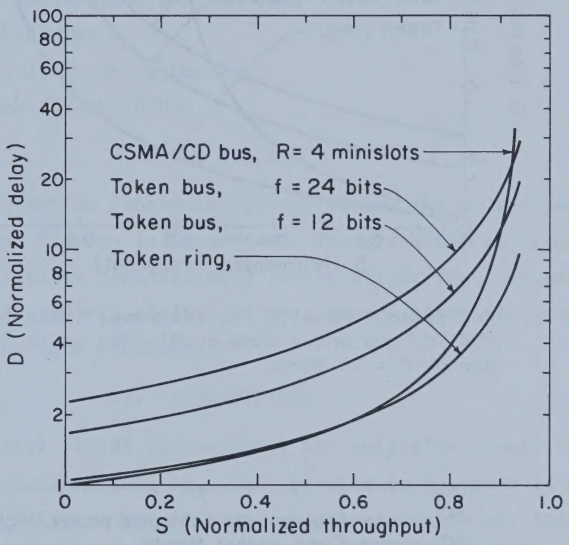


Fig. 1. Performance results for the token ring, token bus with f = 12 and 24 bits, and CSMA/CD bus with R = 4 minislots where C = 1 MBPS and y = 1000 bits.

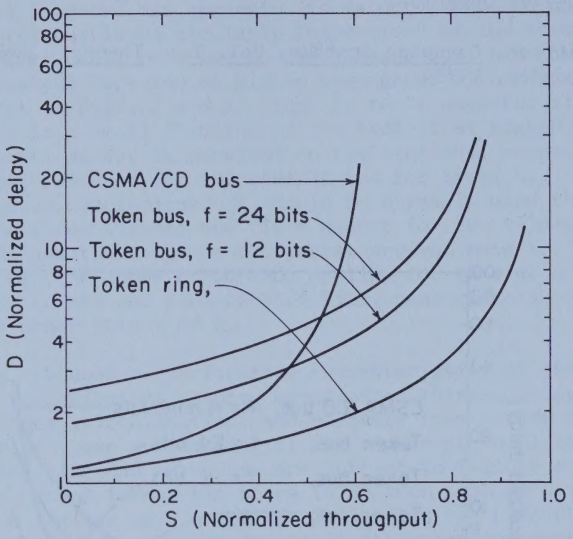


Fig. 2. Performance results for the ring, token bus, and CSMA/CD bus under same conditions as in Fig. 1 except C = 10 MBPS.

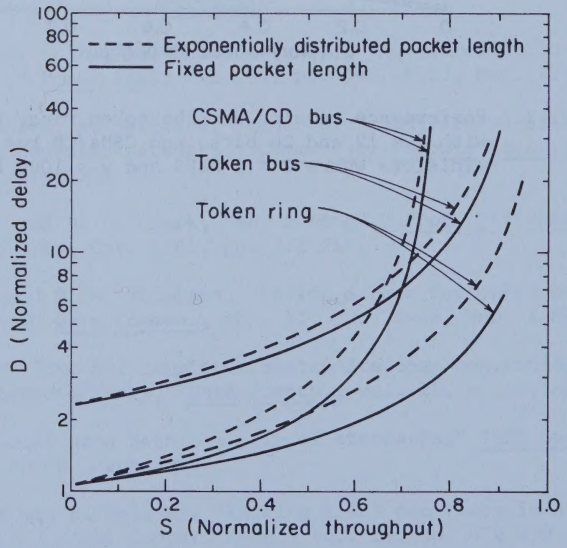


Fig. 3. Performance results for the token ring, token bus with f = 24 bits, CSMA/CD bus with R = 4 minislots, C = 5 MBPS, fixed packet length with y = 1000 bits, and exponentially distributed packet length with $\overline{y}$ = 1000 bits.

A FAST ALGORITHM TO OPTIMIZE THE MULTIPLIER COEFFICIENTS OF CERTAIN CLASSES OF DIGITAL FILTERS

MEHMET TOY
AT&T BELL LABORATORIES
West Long Branch, New Jersey 07740

PAUL M. CHIRLIAN
ELECTRICAL ENGINEERING AND COMPUTER
SCIENCE DEPARTMENT
Stevens Institute of Technology
Hoboken, New Jersey 07030

A B S T R A C T

A new optimization procedure for optimizing the multiplier coefficients of digital filters is presented. The procedure is simple to implement and requires comparatively little effort for computation. It can be applied to IIR and FIR digital filters whose multiplier coefficient errors affect the zero frequency.

I. INTRODUCTION

When a digital filter is realized, the multiplier coefficients cannot, in general, be realized exactly. This results in an error in the frequency response of the filter. For instance, passband ripple may increase; bandwidth may increase or decrease; the attenuation in stopband regions may decrease. In addition to these errors, the value of the transfer function at zero frequency may shift. This will not occur in certain structures. For instance, the digital ladder whos multipliers correspond to the element of an analog ladder filter on a one-to-one basis [17], does not show the level shift [1].

The zero frequency error for wave digital filters hass been studied [11, 12] by Van Haften and Chirlian. However, the zero frequency level shift in wide classes of digital filters has not been studied and it has not been related to the optimization of multiplier coefficients.

This paper is based upon a portion of dissertation that has been accepted in partial fulfillments of the requirements of the Ph.D degree of the Stevens Institute of Technology. The work was performed while M. Toy was with the Stevens Institute of Technology.

The level shift at zero frequency itself may not be important as long as it stays within a tolerable range, but our experimental results show that one can improve the performance of many digital filters at other frequencies by appropriately reducing the error at zero frequency.

II. DEVELOPMENT OF THE PROCEDURE

Let us consider a doubly terminated analog filter of Fig. 1. Its transfer function is

$$H(s) = V_o(s)/V_i(s) \tag{1}$$

At zero frequency $H(j\omega)$ becomes

$$H_o = H(j\omega)\Big|_{\omega = 0} = R_L/(R_S + R_L) \tag{2}$$

If the terminating resistances are changed by ΔR_S and ΔR_L, then the zero frequency error is

$$\varepsilon_o = \Delta H_0 = \frac{(R_S\Delta R_L - R_L\Delta R_S)}{(R_S + R_L)^2\left[1 + \dfrac{(\Delta R_S + \Delta R_L)}{(R_S + R_L)}\right]} \tag{3}$$

If $\left|\dfrac{(\Delta R_S + \Delta R_L)}{(R_S + R_L)}\right| << 1$, then

$$\varepsilon_o = (R_S\Delta R_L - R_L\Delta R_S)/(R_S + R_L)^2 \tag{4}$$

$|\varepsilon_o|$ can be reduced by adjusting either (or both) of the terminating resistances. For instance, if R_S is varied then if $\varepsilon_o > 0$ then R_S must be increased and if $\varepsilon_o < 0$ then R_S must be decreased. Hence, assuming that $\Delta R_L = 0$, we have

$$\Delta R_S = (R_S + R_L)^2\varepsilon_o/R_L \tag{5}$$

The effect of finite multiplier wordlength on response has been studied for recursive and nonrecursive digital filters [13-16]. A number of optimization techniques have been suggested to reduce the error resulting from finite wordlength multiplier coefficients [2-10]. The procedure that we propose here is simpler to implement than those suggested previously and can be used to reduce passband errors in both IIR and FIR digital filters. This procedure appears promising for those digital filters whose zero frequency response is related to its multiplier coefficients.

The procedure that we shall present here cannot be rigorously derived. However, it converges rapidly so that the user can quickly determine if it works. Now, let us consider the following digital filter. Its transfer function is assumed to be exactly obtained by applying the bilinear z-

transform to the analog filter transfer function.

$$H(z) = \sum_{i=0}^{M} a_i z^{-i} \Big/ \left(1 + \sum_{i=1}^{M} b_i z^{-i}\right) \tag{6}$$

where a_i and b_i are infinite precision coefficients. If infinite precision coefficients, a_i and b_i, are represented with a finite number of bits, then the transfer function in (6) becomes

$$H'(z) = \sum_{i=0}^{M} a_i' z^{-i} \Big/ \left(1 + \sum_{i=1}^{M} b_i' z^{-i}\right) \tag{7}$$

$$\text{where } \begin{matrix} a_i' = a_i + \Delta a_i \\ b_i' = b_i + \Delta b_i \end{matrix} \quad , \; i = 1, 2, \ldots, M \tag{8}$$

Δa_i and Δb_i are the quantization errors.

H'(z) differs than H(z) at all frequencies as well as at zero frequency, i.e., z = 1. The error function is

$$\varepsilon(z) = H'(z) - H(z) \tag{9}$$

At zero frequency, we write

$$\varepsilon_o = H_o' - H_o \tag{10}$$

$$H_0 = H(z)\Big|_{z=1} = \sum_{i=0}^{M} a_i \Big/ \left(1 + \sum_{i=1}^{M} b_i\right) \tag{11}$$

$$H_0' = H'(z)\Big|_{z=1} = \sum_{i=0}^{M} a_i' \Big/ \left(1 + \sum_{i=1}^{M} b_i'\right) \tag{12}$$

Substituting (11) and (12) into (10), we obtain

$$\varepsilon_o = \frac{\left(1 + \sum_{i=1}^{M} b_i\right)\left(\sum_{i=0}^{M} \Delta a_i\right) - \left(\sum_{i=1}^{M} \Delta b_i\right)\left(\sum_{i=0}^{M} a_i\right)}{\left(1 + \sum_{i=1}^{M} b_i\right)\left(1 + \sum_{i=1}^{M} (b_i + \Delta b_i)\right)} \tag{13}$$

$|\varepsilon_o|$ can be reduced by adjusting the variables involved in (13). If the procedure is to work, then the adjustment which reduces $|\varepsilon_o|$ must also decrease the errors at other frequencies. It will be shown experimentally that the reducing $|\varepsilon_o|$ by adjusting R_S leads to a reduction on passband and stopband errors.

A change in R_S will change the transfer function of the analog prototype filter. This in turn modifies the transfer function of the digital

filter. This then changes the multiplier coefficients. One might hope that the new digital filter would have zero error at zero frequency. However, this will, in general, not be true because the quantization errors in the new filter will not be related in any simple way to those of the original filter. The change in R_S, ΔR_S, that is used to recompute the new filter should take all these factors into account.

III. CALCULATION OF THE OPTIMIZED TRANSFER FUNCTION COEFFICIENTS AND ΔR_S

Consider the all-pole filter prototypes illustrated in Fig. 2. (a) and (b). In the s-domain, we have

$$H(s) = 1/(d_1 s + d_2 s^{M-1} + .. + d_{M+1}) \tag{14}$$

where d_i is a function of R_S, R_L, ℓ_i and c_i. Replacing s by $2f_s(1-z^{-1})/(1+z^{-1})$ in (14), we have

$$H(z) = (1+z^{-1})^M / \sum_{i=0}^{M} k_{i+1} z^{-i} \tag{15}$$

where k_i is an M^{th} order polynomial in f_s whose coefficients are in terms of R_S, R_L, ℓ_i and c_i. The details of the derivation are given in [1]. From (15) we obtain

$$H(z) = \sum_{i=0}^{M} \frac{\alpha_i}{k_1} z^{-i} \Big/ \left(1 + \sum_{i=1}^{M} \frac{k_{i+1}}{k_1} z^{-i}\right) \tag{16}$$

where

$$\alpha_i = M!/i!(M-i)! \tag{17}$$

Then

$$H(z) = \sum_{i=0}^{M} a_i z^{-i} \Big/ \left(1 + \sum_{i=1}^{M} b_i z^{-i}\right) \tag{18}$$

where

$$a_i = \alpha_i/k_1 \quad , \quad b_i = k_{i+1}/k_1 \tag{19}$$

a_i and b_i are infinite precision number. If finite precision is used for a_i and b_i, then H(z) becomes H'(z) given in (7),

Similar expressions can be obtained for the prototype in Fig. 2.(c). [1]. The disctete transfer function can be expressed as in (18) with the definitions of (19). Here α_i is a function of R_L, ℓ_i and c_i.

It can be shown that k_i is a linear function of ΔR_S[1]. Thus, if R_S changes by ΔR_S, then k_i becomes $k_i + \Delta k_i$. Now Δk_i is proportional to ΔR_S, so that we can write

$$\Delta k_i = K_i \Delta R_s \tag{20}$$

The new transfer function H(z) can be obtained from (16) replacing k_i by $k_i + \Delta k_i$. Hence,

$$\hat{H}(z) = \sum_{i=0}^{M} \frac{a_i}{(k_i+\Delta k_i)} z^{-i} \Big/ \left(1 + \sum_{i=1}^{M} \frac{(k_i+\Delta k_i)}{(k_i+\Delta k_l)} z^{-i}\right) \tag{21}$$

Now

$$\hat{H}(z) = \frac{\sum_{i=0}^{M} \hat{a}_i z^{-i}}{1 + \sum_{i=1}^{M} \hat{b}_i z^{-i}} \tag{22}$$

where

$$\hat{a}_i = a_i/(k_1 + \Delta k_1) \ , \ \hat{b}_i = (k_{i+1} + \Delta k_{i+1})/(k_1 + \Delta k_1) \tag{23}$$

Now $\hat{a}_i$ and $\hat{b}_i$ are infinite precision coefficients. If we represent them with finite number of bits, then the transfer function in (22) becomes

$$\hat{H}'(z) = \sum_{i=0}^{M} \hat{a}'_i z^{-i} \Big/ \left(1 + \sum_{i=1}^{M} \hat{b}'_i z^{-i}\right) \tag{24}$$

where

$$\hat{a}'_i = \hat{a}_i + \Delta\hat{a}_i \ , \ \hat{b}'_i = \hat{b}_i + \Delta\hat{b}_i \tag{25}$$

Here a_i and b_i are the quantization errors.

The zero frequency error for the new transfer function is

$$\hat{\varepsilon}_o = \hat{H}'_0 - \hat{H}_0 \tag{26}$$

where

$$\hat{H}'_0 = \hat{H}'(z)\Big|_{z=1} \quad \text{and} \quad H_0 = H(z)\Big|_{z=1}$$

Setting $\hat{\varepsilon}_o$ equal to zero, we obtain [1].

$$\frac{1}{\Delta R_S(1)} = -\frac{K_1}{k_1} - \frac{\left(\sum_{i=0}^{M} a_i\right)\left(\sum_{i=1}^{M+1} K_i\right)}{k_1\left[\left(\sum_{i=0}^{M} a_i\right)\left(\sum_{i=1}^{M} \Delta\hat{b}_i\right) - \left(1 + \sum_{i=1}^{M} b_i\right)\left(\sum_{i=0}^{M} \Delta\hat{a}_i\right)\right]} \tag{27}$$

We do not know the new values of the quantization errors until the new filter is realized and we cannot determine the transfer function of the new filter until the quantization errors are evaluated. Thus, it seems that we cannot use (27) to find the deviation of the source resistance from its original value to make zero frequency error zero. However, we resolve this problem by guessing at the quantization errors. In particular we choose

$$\Delta\hat{a}_i = \Delta a_i \quad , \quad \Delta\hat{b}_i = \Delta b_i \tag{28}$$

The validity of such a guess may be questioned. However, the value of ΔR_S obtained is only used as a starting point in an iteration procedure. Thus, some error can be tolerated. Before we discuss the search procedure to find ΔR_S we shall consider the minimum multiplier wordlength for which the procedure can be applied.

IV. FAILING WORDLENGTH, N_F

Our computer simulation results indicate that the procedure will not converge if the multiplier wordlength falls below a value N_F which we call the failing wordlength. In such cases, no significant change in the transfer occurs in the multiplier coefficients are adjusted in accordance with the procedure. Experimentally [1] we have found that $N_F = 3$ for FIR and IIR direct-structured filters.

V. A SEARCH TECHNIQUE FOR ΔR_S

The value of ΔR_S obtained from (4) or (27) is usually too large. We have been able to find acceptable values of ΔR_S using the following algorithm.

i) Use the experimentally determined value of N_F and compute H_0' for $N > N_F + 1$

ii) Obtain ΔR_S from either (4) for the IIR direct-structured filters or from (27) for all other IIR and FIR digital filters to which the procedure is applicable, and compute $\hat{H}_0'$.

iii) Consider $\hat{H}_0'$ obtained in (ii):

a) If $\hat{H}_0'$ is equal to H_0, then stop the iteration.

b) If the zero frequency error is increased, $|\hat{\varepsilon}_o| > |\varepsilon_o|$, and H_0 is not within the interval $[\hat{H}_0' , H_0']$, then reverse the sign of ΔR_S and try it, and choose ΔR_S as in (c), i.e., $\hat{H}_0'$ is in the region I in Fig. 3.

c) If either $|\hat{\varepsilon}_o| > |\varepsilon_o|$ or the interval $[\hat{H}_0' , H_0']$ contains H_0, then decrease ΔR_S by half, i.e., $\hat{H}_0'$ is in the regions II, III, IV in Fig. 3.

iv) Compute $\hat{H}_0'$ using ΔR_s obtained in (iii)

a) If the interval [$\hat{H}_0'$ obtained in this step, H_0' obtained in (iii)] contains H_0, then choose ΔR_s as (1/2) [(Upper zero frequency level) - (Lower zero frequency level)] + Lower zero frequency level, where upper and lower zero frequency levels are either the new $\hat{H}_0'$ level obtained in this step or $\hat{H}_0'$ level obtained in (iii).

b) If the interval [$\hat{H}_0'$ obtained in this step, H_0' obtained in (iii)] does not contain H_0, then decrease ΔR_s by half till the interval [$\hat{H}_0'$ new, $\hat{H}_0'$ obtained in (iii)] contains H_0. Then choose ΔR_s as in (a).

v) Compute the new $\hat{H}_0'$ using ΔR_s obtained in (iv). Either the interval [$\hat{H}_0'$ obtained in this step, $\hat{H}_0'$ obtained in (iv)] or the interval [$\hat{H}_0'$ obtained in this step, $\hat{H}_0'$ obtained in (iii)] contains H_0. Pick up the interval contains H_0, which is the solution and choose ΔR_s as in (iv)-a.

vi) Repeat step (v) till the adjustment of ΔR_s does not create a change of the multipliers which have been obtained in the previous step.

After step (vi), if still there is an error at zero frequency, reverse the sign of ΔR_s obtained in the last step and try it. Note that we assumed the sign of ΔR_s in (27) and (5).

In general, at each step, the stability should be tested, any ΔR_s value which leads to the unstability must be discarded, and the maximum passband and stopband deviations of the transfer function should be computed. Usually, instability does not result. The details of the algorithm is given in [1].

VI. EXPERIMENTAL RESULTS

We have applied the procedure to a number of IIR and FIR filters [1], i.e., block filters, scalar ladder and lattice filters, etc. In each case, there is substantial improvements in the passband responses. Insignificant improvements in the stopband responses can also be observed. An experimental result is given in Figure 4.

VII. CONCLUSION

A new optimization procedure for reducing the effects of multiplier coefficient quantization has been presented. Although this procedure is heuristic and has not been rigorously derived, it still appears attractive.

The procedure can be simply implemented, requires a few iteration steps, and is applicable to wide classes of digital filters.

Experimental results indicate that for certain classes of digital filters, significant improvement in the response can be obtained. For instance, in a seventh order direct-structured filter, the passband error has been reduced from almost 14 dB to the design specifications.

REFERENCES

1. M. Toy, "Block Digital Ladder Filters and Finite Multiplier Wordlength Errors in Digital Filters," Ph.D Thesis, Stevens Institute of Technology, 1982.
2. E. Evenhaus, "On the Design of Digital Filters with Coefficients of Limited Wordlength," IEEE Trans. Audio Electroac., Vol. AU-20, pp. 206-212, August 1972.
3. K. Steiglitz, "Designing Short Word Recursive Digital Filters," Proc. 9th Annual Allerton Conf. on Circuit and Systems Theory, pp. 778-788, October 6-8, 1971.
4. M. Suk and S. K. Mitra, "Computer-Aided Design of Digital Filters with Finite Wordlengths," IEEE Trans. on Audio and Electroac., Vol. AU-20, No. 5, pp. 356-363, December 1972.
5. C. Charalambous and M. J. Best, "Optimization of Recursive Filters with Finite Wordlengths," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-22, No. 6, pp. 424-431, December 1974.
6. F. Brglez, "Digital Filter Design with Short Wordlength Coefficients," IEEE Trans. on Circuits and Systems, Vol. CAS-25, No. 12, Dec. 1978.
7. K. A. Owenier, "A General Optimization Procedure for Wave Digital Filters with Reduced Number of Multipliers," Proc. 1976, IEEE Symp. Circuits and Systems, pp. 461-464.
8. D. S. K. Chan and L. R. Rabiner, "Analysis of Quantization Errors in the Direct Form for Finite Impulse Response Digital Filters," IEEE Trans. Audio Electroac., Vol. AU-21, pp. 354-366, August 1973.
9. F. Brglez, "Sampling Rate and Wordlength Minimization in FIR Digital Filters," Proc. 1978, IEEE Symp. Circuits and Systems, pp. 1038-1039.
10. V. B. Lawrence and A. C. Salazar, "Finite Precision Design of Linear-Phase FIR Filters," The Bell Syst. Tech. J., Vol. 59, No. 9, pp. 1575-1598, November 1980.
11. Daniel L. Van Haften, "An Analysis of Errors in Wave Digital Filters," Ph.D Thesis, Stevens Institute of Technology, 1977.

12. D. L. Van Haften, P. M. Chirlian, "An Analysis of Errors in Wave Digital Filters," IEEE Trans. Circuits and Systems, Vol. CAS-28, No. 2, pp. 154-160, February 1981.
13. J. B. Knowles and M. Olcayto, "Coefficient Accuracy and Digital Filter Response," IEEE Trans. Circuit Theory, Vol. CT-15, pp. 31-41, March 1968
14. R. E. Crochiere, "A New Statistical Approach to the Coefficient Word-length Problem for Digital Filters," IEEE Trans. Circuits and Systems, Vol. CAS-22, pp. 190-196, March 1975.
15. A. Gersho, B. Gopinath and A. M. Odlyzko, "Coefficient Inaccuracy in Transversal Filtering." The Bell Syst. Tech. J., Vol. 58, No. 10, pp. 2301-2316, December 1979.
16. L. T. Burton, "Low Sensitivity Digital Ladder Filters," IEEE Trans. Circuits and Systems, Vol. CAS-22, No. 3, March 1975.

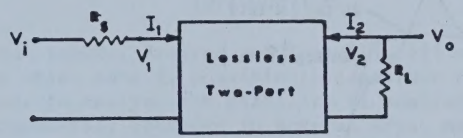


FIGURE 1 - A doubly terminated lossless two-port

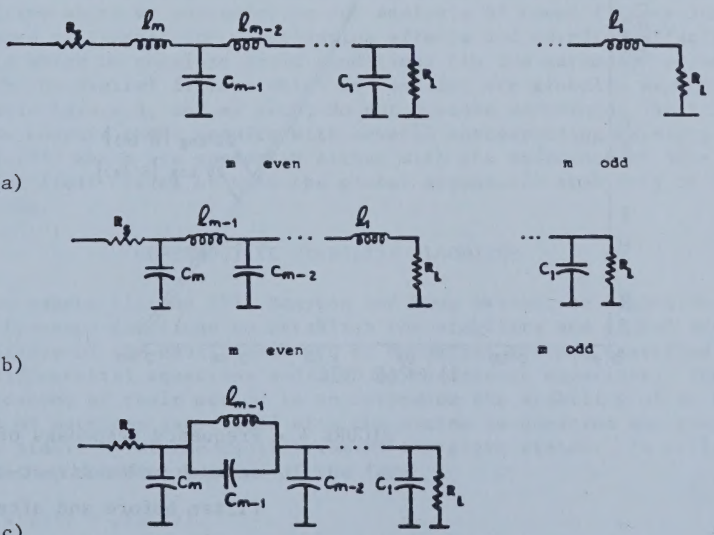


FIGURE 2 - RLC Ladder Prototypes

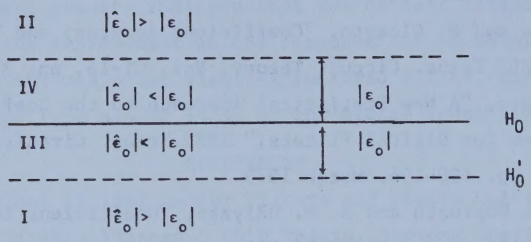


FIGURE 3

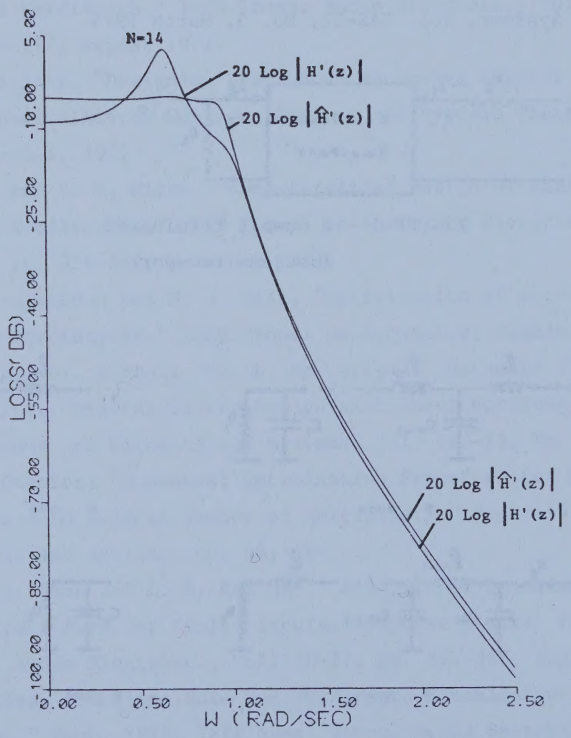


FIGURE 4 - Frequency responses of a seventh order direct-structured filter before and after the optimization, i.e., N = 14 and N_S (Number of iteration steps) = 11

STABILITY ANALYSIS OF FIXED-POINT DIGITAL FILTERS USING COMPUTER GENERATED LYAPUNOV FUNCTIONS: WAVE AND LATTICE FILTERS

KELVIN T. ERICKSON
Fisher Controls Co.
Marshalltown, Iowa 50158

ANTHONY N. MICHEL
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

We demonstrate the applicability of the constructive stability algorithm of Brayton and Tong in the stability analysis of fixed-point digital filters. In the present paper, we consider wave and lattice digital filter structures. We compare our results with existing ones which deal with the stability of fixed-point digital filters. Several of the present results are new while some of them constitute improvements over existing results. It is emphasized that whereas the existing results are obtained by several diverse methods, the present results are determined by one unified approach.

INTRODUCTION

In two recent papers, Brayton and Tong [1], [2] established some significant results which make it possible to construct computer generated Lyapunov functions to analyze the stability of nonlinear systems. These results were subsequently extended in several ways, making it possible to estimate the domain of attraction of an equilibrium [3] and to apply the constructive algorithm to high-dimensional systems [4].

In this paper, we apply the constructive algorithm of Brayton and Tong to the stability analysis of second order wave and lattice fixed-point digital filters. Direct form and coupled form filters are considered in [5a]. Nonlinearities which we encounter in our analysis of these filters include several types of fixed-point quantization effects and overflow effects. The results which we obtained yield conditions (in the parameter plane) under which the digital filters which we consider are globally asymptotically stable (g.a.s.), and as such, do not possess zero-input limit cycles. We compare these results with several corresponding existing results ([6]-[9]) which are concerned either with the existence or nonexistence of limit cycles or with the global asymptotic stability of discrete systems.

CONSTRUCTIVE STABILITY ALGORITHM

In two papers [1] and [2], Brayton and Tong present an algorithm to construct Lyapunov functions to establish the stability and global asymptotic stability of the equilibrium $x=0$ of dynamical systems described by ordinary differential equations and also by difference equations. The basic philosophy of their method is to determine the stability of an appropriate set of matrices associated with the system in question and then deduce the stability of the equilibrium of the given system. To utilize this algorithm, consider systems of the form

$$x(k+1) = g[x(k)]. \tag{1}$$

We rewrite the system of equations (1) as

$$x(k+1) = M(x(k))x(k) \tag{2}$$

where $M(x(k))$ is chosen so that $M(x(k))x(k) = g[x(k)]$. For every $x \in R^n$, $M(x)$ will be a real $n \times n$ matrix. If we let $\underline{M}$ denote the set of all matrices M, obtained by varying x in $M(x)$ over all allowable values, then we can rewrite (2) equivalently as

$$x(k+1) = M_k x(k), \quad M_k \in \underline{M}. \tag{3}$$

Brayton and Tong [1], [2] show that the equilibrium $x=0$ of (3) is stable (globally asymptotically stable) if the set of matrices $\underline{M}$ is stable (asymptotically stable). In the following, we give a brief summary of the results of Brayton and Tong.

We call a set $\underline{A}$ of $n \times n$ real matrices stable if for every neighborhood of the origin $U \subset R^n$, there exists another neighborhood of the origin $V \subset R^n$ such that for every $M \in \underline{A}'$, we have $MV \subseteq U$. Here, $\underline{A}'$ denotes the multiplicative semigroup generated by a set $\underline{A}$ and $MV = \{u \in R^n: \ u = Mv, \ v \in V\}$.

In [1] it is shown that if a set $\underline{A}$ of $n \times n$ matrices is stable, then there exists a vector norm $|\cdot|_w$ such that $|Mx|_w \leq |x|_w$ for all $M \in \underline{A}$ and $x \in R^n$. The vector norm $|\cdot|_w$ defines a Lyapunov function for $\underline{A}$, i.e., it defines a function v with the property

$$v(Mx) \leq v(x), \quad \text{for all} \quad M \in \underline{A} \quad \text{and} \quad x \in R^n.$$

Next, we call a set of matrices $\underline{A}$ asymptotically stable if there exists a number $\rho > 1$ such that $\rho\underline{A}$ is stable. (The set $\rho\underline{A}$ is obtained by multiplying every element of $\underline{A}$ by ρ.) Thus, if $\rho\underline{A}$ is stable, then there exists a vector norm $|\cdot|_w$ such that

$$|Mx|_w < |\rho Mx|_w \leq |x|_w \quad \text{for all} \quad M \in \underline{A} \quad \text{and} \quad x \in R^n.$$

The vector norm $|\cdot|_w$ defines a Lyapunov function with the property

$$v(Mx) < v(x), \quad \text{for all} \quad M \in \underline{A} \quad \text{and} \quad x \in R^n.$$

Thus, given the system (3), if we can show that $\underline{M}$ is asymptotically stable, then we can conclude that the system (3) is globally asymptotically stable, by Lyapunov stability theory (see, e.g., [10]).

In [1] and [2], a constructive algorithm is given to determine whether a finite set of real matrices is either stable, by constructing the Lyapunov function, or unstable by determining that the vector norm Lyapunov function does not exist.

The set of matrices $\underline{M}$ given in (3) consists in general of an infinite number of matrices. However, in [1] it is shown that we need only work with the extreme matrices of the set, $E(\underline{A})$, to get an indication of the stability of $\underline{A}$. In applications, $E(\underline{A})$ is frequently a finite set, which is used to analyze the stability of the infinite set, $\underline{A}$.

APPLICATION OF THE CONSTRUCTIVE ALGORITHM TO THE STABILITY ANALYSIS OF DIGITAL FILTERS

Nonlinearities in Digital Filters

In fixed-point digital filters, the representation of signals must necessarily have finite precision. Multiplications and additions performed in the digital filter generally lead to an increase in the wordlength required for the result of the operation. In a recursive digital filter, a

wordlength reduction of the result is necessary to prevent the wordlength of the signals from increasing indefinitely. A quantization nonlinearity is produced when the result of a multiplication is reduced in wordlength. An overflow nonlinearity occurs when the sum of two numbers is modified to fall back within the representable range. These two types of nonlinearities are well described in the literature and, therefore, are not discussed here. The characteristics of the quantization and overflow nonlinearities that we consider are shown in Figure 1. Other types of overflow nonlinearities are considered in [5].

These nonlinearities may be viewed as belonging to a sector [k,K]. Thus, if $f(\cdot)$ denotes a given nonlinearity, then

$$k\sigma^2 \leq \sigma f(\sigma) \leq K\sigma^2 \quad \text{for all} \quad \sigma \in R,$$

where k and K are constants such that $-\infty < k \leq K < \infty$. Under this assumption, we view the quantization nonlinearities as belonging to the sector $[0, k_q]$ where

$$k_q = \begin{cases} 1 & \text{truncation} \\ 2 & \text{roundoff.} \end{cases} \tag{4}$$

Overflow is represented as belonging to the sector $[k_o, 1]$ where

$$k_o = \begin{cases} 0 & \text{saturation} \\ -1 & \text{two's complement.} \end{cases} \tag{5}$$

When the quantization and overflow functions are executed simultaneously, then the combined nonlinear operation is represented as belonging to the sector $[k_o, k_q]$.

General Digital Filter

Since we consider only second order digital filters, the matrix M in (2) may be written as

$$M(x(k)) = \begin{bmatrix} a(x(k)) & b(x(k)) \\ c(x(k)) & d(x(k)) \end{bmatrix}.$$

We assume that the elements of M satisfy the inequalities

$$a_1 < a(x(k)) < a_2 \qquad c_1 < c(x(k)) < c_2$$

$$b_1 < b(x(k)) < b_2 \qquad d_1 < d(x(k)) < d_2$$

where a_i, b_i, c_i and d_i, i=1,2 are constants. Then, the extreme matrices of set $\underline{M}$ in (3) are obtain as

$$E(\underline{M}) = \left\{ \begin{bmatrix} a_i & b_j \\ c_k & d_\ell \end{bmatrix}, \; i,j,k,\ell = 1,2 \right\}. \tag{6}$$

We need only determine that $E(\underline{M})$ is asymptotically stable to determine that the digital filter described by (1) is globally asymptotically stable. If the set $\underline{M}$ is unstable, then we can draw no conclusion about the stability of the filter. Since the constructive algorithm is used to show that the equilibrium x=0 of a given digital filter (1) is globally asymptotically stable, then in particular, no limit cycles will exist in this digital filter.

To implement the constructive algorithm, we evaluate the set of extreme matrices (6) at each point in a sufficiently fine grid in the param-

eter plane. We used a grid of 0.02 units for the results presented here. The computer programs that implement the constructive algorithm are described in detail in [12].

Wave Digital Filter

Since wave digital filters are a class of filters, we consider here only a general second order lowpass LC filter (Fig. 2). Following the usual synthesis procedures, the recursive part of the resulting structure with two quantizers, is formulated as in Figure 3a. Here, the quantizations and overflow nonlinearities are applied at the states of the filter. When the state equations are written in the form of (2), a set of extreme matrices is recognized since the nonlinearities belong to sectors. It can be shown ([5b], [12]) that the set of extreme matrices for the wave filter with two quantizers is given by

$$E(\underline{M}) = \left\{ \begin{bmatrix} (-1-m_1(2+m_2))\alpha_i & m_2\alpha_i \\ (-m_1(2+m_2+m_3))\beta_j & (1+m_2+m_3)\beta_j \end{bmatrix} \quad ,i,j = 1,2 \right\}, \tag{7}$$

Where $m_1 = (-2a)/(1+2a)$, $m_2 = (-1)/(1+a+b+2ab)$, $m_3 = m_2(1+a)$, $a = C/T$, and $b = L/T$. The constants $\alpha_1 = \beta_1 = k_o$ and $\alpha_2 = \beta_2 = k_q$, where k_q and k_o are defined by (4) and (5), respectively. Thus, for each point (a,b) in the parameter plane, there are four extreme matrices. If the overflow nonlinearities are absent, the set of extreme matrices is the same as for the filter with saturation overflow nonlinearities.

In [5b] we also consider structures with three quantizers (Fig. 3b). A discussion of this structure is omitted here, due to space limitations.

Lattice Digital Filter

The linear recursive part of a second order lattice digital filter is shown in Fig. 4. We consider two structures for the nonlinear filter. Quantization and overflow can be applied at the states of the filter. This structure has two quantizers and has received attention by several authors (Fig. 5). We also consider quantization after each multiplication and overflow after each addition. This structure with three quantizers (Fig. 6) is a more realistic implementation of the actual filter using a fixed-point microprocessor.

The set of extreme matrices for the lattice filter with two quantizers can be shown to be equal to

$$E(\underline{M}) = \left\{ \begin{bmatrix} -k_1\alpha_i & -k_2\alpha_i \\ (1-k_1^2)\beta_j & -k_1k_2\beta_j \end{bmatrix} \quad ,i,j = 1,2 \right\} \tag{8}$$

where $\alpha_1 = \beta_1 = k_o$ and $\alpha_2 = \beta_2 = k_q$. Thus, for each point (k_1,k_2) in the parameter plane, there are four extreme matrices. If the overflow nonlinearities are absent, the set of extreme matrices is the same as for the filter with saturation overflow nonlinearities.

The set of extreme matrices for the lattice filter structure with three quantizers is given in [5b]. It is not presented herein, due to space limitations.

COMPARISON OF STABILITY RESULTS BY THE CONSTRUCTIVE ALGORITHM WITH EXISTING RESULTS

Wave Filter

To apply the constructive algorithm to the particular wave filter, we

use the extreme matrices determined by (7). When truncation quantizers are used with either type of overflow, the constructive algorithm shows that this filter is globally asymptotically stable in the region $0 < a \leq 100$ and $0 < b \leq 100$. We did not try larger values of a and b since the run times of the program increased significantly as a and b are increased in value. However, this region does cover any reasonable values of these parameters, since larger values of a and b imply that the sampling frequency of the filter is fairly high compared to the cutoff frequency of the filter. Thus, our results agree essentially with those of Fettweis and Meerkötter [6], [7] who show that this filter is free of limit cycles if truncation quantization and any overflow are applied at the states of the filter.

For this filter with two roundoff quantizers, the region in the parameter plane where the filter is globally asymptotically stable are shown in Fig. 7. The region where a particular filter is globally asymptotically stable is indicated as the unhatched region within the first quadrant. The horizontally hatched region indicates the region where the set of extreme matrices is unstable because one of the eigenvalues of an extreme matrix has magnitude greater than one. Vertical hatching indicates the remainder of the region where the set of extreme matrices is unstable, and thus, where we can draw no conclusion about the stability of the filter. For this filter with two roundoff quantizers and no overflow, the constructive algorithm obtains a less conservative stability region than the application of the absolute stability criterion of Jury and Lee [8] (the boundary of which is indicated by the dashed line in Fig. 7).

The remaining results obtained by the constructive algorithm for this filter when roundoff quantization and overflow are considered together (depicted in Figs. 8 and 9), appear to be new.

Lattice Filter (Two Quantizers)

To apply the constructive algorithm to the lattice filter with two quantizers, we use the extreme matrices determined by (8). When truncation quantizers are used with either type of overflow, the constructive algorithm shows that this filter is globally asymptotically stable when the parameters k_1 and k_2 are withen the unit square in the k_1-k_2 parameter plane. The corresponding linear filter is also globally asymptotically stable within this region. Gray [9] also shows that this nonlinear filter is free of zero-input limit cycles in this region.

For roundoff quantizers, the regions in the parameter plane where the filter is globally asymptotically stable are shown in Figs. 10, 11 and 12. Hatching signifies the same meaning as before. These regions are symmetric about the k_2 axis. These results appear to be new. Application of the absolute stability criterion of Jury and Lee [8] yields no region where this filter is globally asymptotically stable.

For further results, involving lattice filter structures with three quantizers, refer to [5b].

CONCLUSION

The wave digital filter and lattice digital filter structures which we have analyzed demonstrate that the Brayton-Tong constructive algorithm is a powerful tool in the stability analysis of fixed-point digital filters. We have obtained some new results for the structures which we analyzed. We also improved upon some of the existing results.

Whereas existing methods of stability analysis are generally different for each particular filter structure, the constructive algorithm allows us to use one method to study the stability of nonlinear digital filter structures. We feel that this method should become a tool to be used in the evaluation of any proposed new filter structure.

In [5a], second order direct form and coupled form filters are analyzed by the Brayton-Tong algorithm. There are many other digital filter structures one might want to analyze by this method. The constructive algorithm could also be applied to higher order filters either directly or by considering the higher order filter as an interconnection of lower order structures (see, e.g., [4], [11]).

REFERENCES

[1] Brayton, R. K. and Tong, C. H., "Stability of Dynamical Systems: A Constructive Approach". IEEE Transactions on Circuits and Systems, CAS-26 (1979): 224-234.

[2] Brayton, R. K. and Tong, C. H., "Constructive Stability and Asymptotic Stability of Dynamical Systems". IEEE Transactions on Circuits and Systems, CAS-27 (1980): 1121-1130.

[3] Michel, A. N., Sarabudla, N. R. and Miller, R. K., "Stability Analysis of Complex Dynamical Systems: Some Computational Methods". Circuits, Systems and Signal Processing, 1 (1982): 171-202.

[4] Michel, A. N., Miller, R. K. and Nam, B. H., "Stability Analysis of Interconnected Systems Using Computer Generated Lyapunov Functions", IEEE Transactions on Circuits and Systems, CAS-29, (1982): 431-440.

[5a] Erickson, K. T. and Michel, A. N., "Stability Analysis of Fixed-Point Digital Filters Using Computer Generated Lyapunov Functions -Part I: Direct Form and Coupled Form Filters". (To appear in IEEE Transactions on Circuits and Systems, CAS-32, January 1985.)

[5b] Erickson, K. T. and Michel, A. N., "Stability Analysis of Fixed-Point Digital Filters Using Computer Generated Lyapunov Functions--Part II: Wave Digital Filters and Lattice Digital Filters". (To appear in IEEE Transactions on Circuits and Systems, CAS-32, January 1985.)

[6] Fettweis, A. and Meerkötter, K., "Suppression of Parasitic Oscillations in Wave Digital Filters". IEEE Transactions on Circuits and Systems, CAS-22 (1975): 239-246.

[7] Fettweis, A. and Meerkötter, K., "Correction to 'Suppression of Parasitic Oscillations in Wave Digital Filters'". IEEE Transactions on Circuits and Systems, CAS-22 (1975): 575.

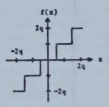
[8] Jury, E. I. and Lee, B. W., "The Absolute Stability of Systems with Many Nonlinearities". Automation and Remote Control, 26 (1965): 943-961.

[9] Gray, A. H., "Passive Cascaded Lattice Digital Filters". IEEE Transactions on Circuits and Systems, CAS-27 (1980): 337-344.

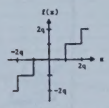
[10] Miller, R. K. and Michel, A. N., Ordinary Differential Equations. New York: Academic Press, 1982.

[11] Michel, A. N. and Miller, R. K., Qualitative Analysis of Large Scale Dynamical Systems. New York: Academic Press, 1977.

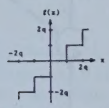
[12] Erickson, K. T., "Stability Analysis of Fixed-Point Digital Filters Using a Constructive Algorithm". Ph.D. Dissertation, Iowa State University, Ames, Iowa, 1983.



(a) Roundoff

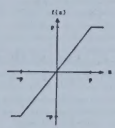


(b) Magnitude truncation

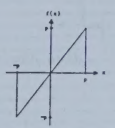


(c) Value truncation

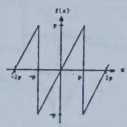
Fixed-point quantization characteristics.
Figure 1



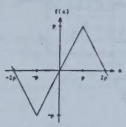
(d) Saturation



(e) Zeroing

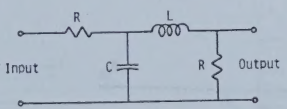


(f) Two's complement

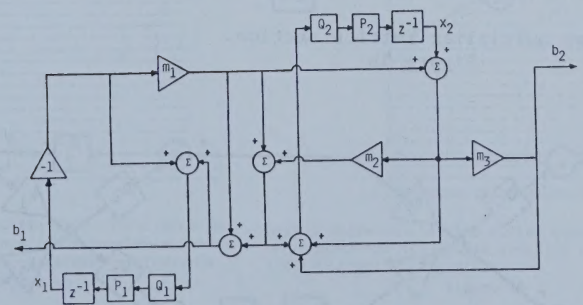


(g) Triangular

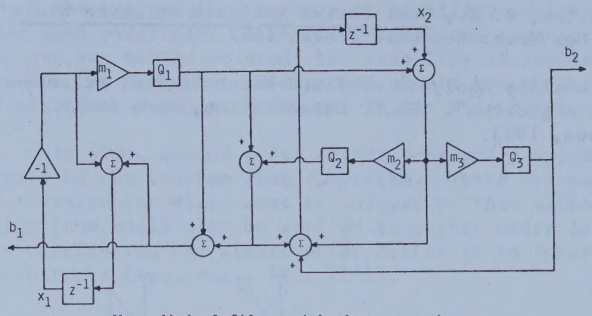
Overflow characteristics.
Figure 1



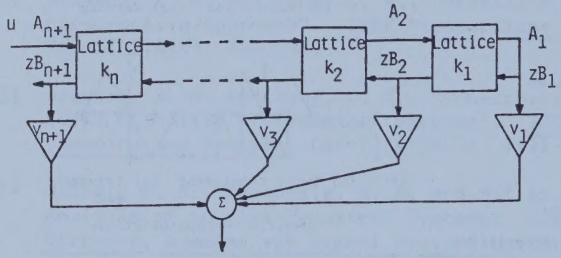
General second order
LC lowpass analog filter.
Figure 2



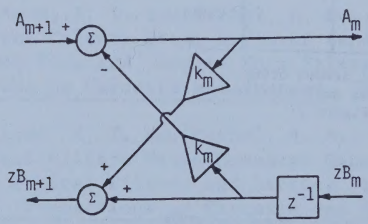
Wave digital filter with two quantizers.
Figure 3a



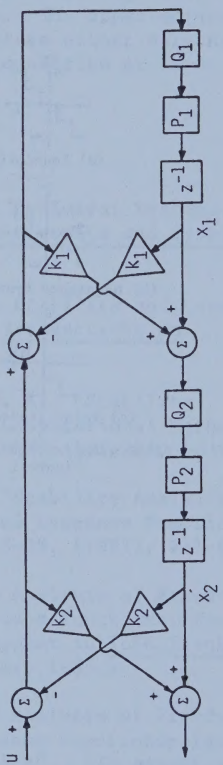
Wave digital filter with three quantizers.
Figure 3b



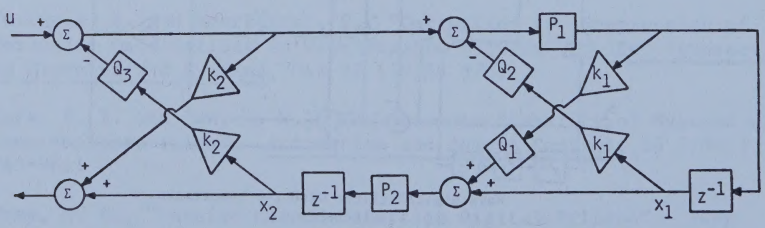
General lattice digital filter structure.
Figure 4a



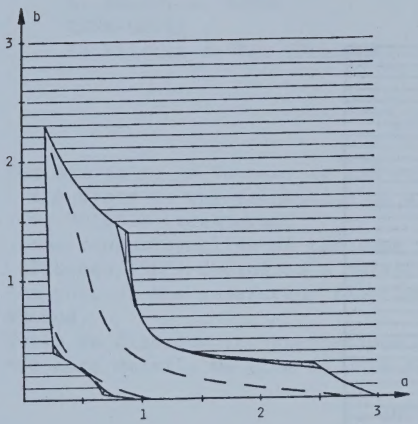
Two multiplier lattice section.
Figure 4b



Lattice digital filter with two quantizers.
Figure 5



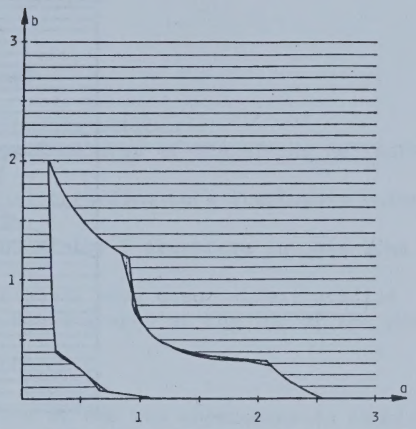
Lattice digital filters with three quantizers.
Figure 6



- - - Boundary by Theorem 4 of [8]

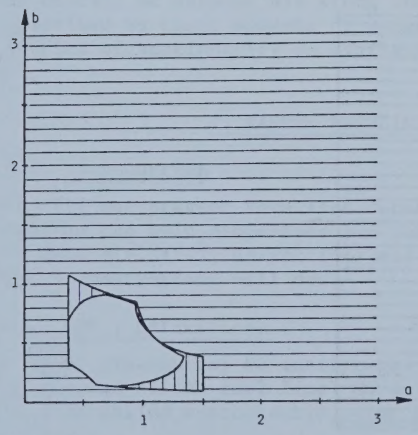
Region where the specific wave filter with two roundoff quantizers and saturation, zeroing or no overflow is g.a.s. by the constructive algorithm.

Figure 7



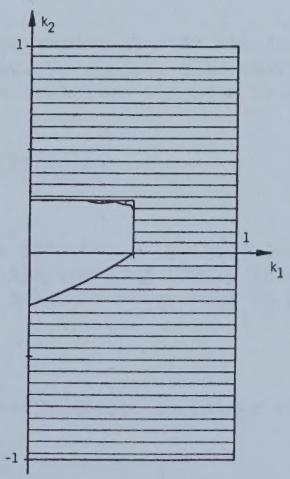
Region where the specific wave filter with two roundoff quantizers and triangular overflow is g.a.s. by the constructive algorithm.

Figure 8



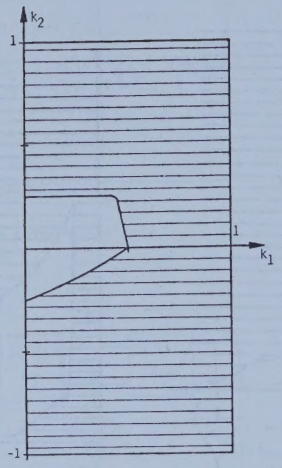
Region where the specific wave filter with two roundoff quantizers and two's complement overflow is g.a.s. by the constructive algorithm.

Figure 9



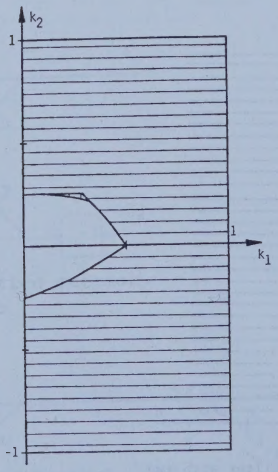
Region where the lattice filter with two roundoff quantizers and triangular overflow is g.a.s. by the constructive algorithm.

Figure 10



Region where the lattice filter with two roundoff quantizers and triangular overflow is g.a.s. by the constructive algorithm.

Figure 11



Region where the lattice filter with two roundoff quantizers and two's complement overflow is g.a.s. by the constructive algorithm.

Figure 12

ISOMORPHISM BETWEEN THE CONTINUOUS SPACES SIGNALS AND DISCRETE SPACES SIGNALS. COMPLEMENTS

C. BOZZO, G. BUENO
CSEE-CETIA
La Valette du Var (France)

ABSTRACT

Our aim is to go further in the relationship between :
. the discrete representation of a signal by mean of transforms as LAGUERRE or POISSON transforms,
. the random sampling of the same signal according to a Stationary Point Process (SPP) obeing to a POISSON law.
The role of the generating function and of the Z transform in this link is showed.
Works on discrete representations of signals have been widely studied.
For more detail, we give an extensive bibliography at the end of the paper.

1. INTRODUCTION

We present briefly here the main results of the two approaches in chapter 2 and 3. In chapter 4 we show the links between them and how to compute easily the POISSON transform coefficients of the correlation function from signal samples randomly obtained. We show also the complete consistency of this result, with the periodic case.

Finaly, we conclude on the lake of unified approach of the two ways of random signals discretizations.

Of course, we suppose all along the paper, that random signals are fully described by their moments of order one and two. We suppose also that properties of stationarity is available and the correlation function belong to L^2.

2. RANDOM SAMPLING - A BRIEF APPROACH

2.1. Introduction

We will not present here in a detailed way the SPP theory. We will just mention the main properties and the link with the sampling theory (random or deterministic), definitions and properties needed further will be given, the demonstrations will not.

2.2. SPP : Définitions

A point process may be interpreted as an ordered sequence of points randomly located on the real line. So we can say :
$\{t_n\}$ is called a stochastic point process if :

let $\{\tau_n\}$ $(n \in Z)$ be a discrete parameter stochastic process associated with a probability space (Ω,F,P), and such that τ_n be finite-valued and non-negative with probability one, so

$$t_n = \begin{cases} \sum_{o}^{n} \tau_k & \text{if } n \geq 0 \\ \tau_o - \sum_{n}^{-1} \tau_k & \text{if } n \leq -1 \end{cases} \qquad (1)$$

$\{t_n\}$ is called a stochastic point process

Furthermore, we will called a stationary non decreasing point process, each $\{t_n\}$ with

- $t_k \leq t_{k+1}$
- statistical properties of the numbers of points in any finite set of intervals are invariante by any translation.

Such definition is particularly interesting if we consider, as BEUTLER and LENEMAN does in [19], the process $L_n(t)$ such that :

$L_n(t) \triangleq$ length of time required for the k^{th} point, after t, to occur.

Note that the distribution functions of $L_n(t)$ provide a direct description of the statistics of samples locations relative to an arbitrary t, which is precisely the information needed for the determination of an auto correlation function.

If we define the distribution function of the $L_n(t)$ process, as

$$G_n(x,t) = P\left[L_n(t) \leq x\right] \tag{2}$$

under stationarity hypothesis, we have :

$$\begin{cases} L_n(t) = L_n \\ G_n(x,t) = G_n(x) \end{cases} \tag{3}$$

as the distribution function of $L_n(t)$ possess convexity and absolute continuity properties [19], we define the density probability function $g_n(x)$ as

$$g_n(x) = \frac{dG_n(x)}{dx} \tag{4}$$

2.3. POISSON sampling process

If we have the $\{\tau_k\}$ sequence such that :

→ τ_k is independent of $\tau_j \quad \forall\, k \neq j$

→ the τ_k sequence is exponentialy distributed

$$\left\{ F(x) = 1 - \exp(\beta x) \right\}$$

One can show that $\{t_i\}$ is POISSON distributed and $g_n(x)$ is

$$g_n(x) = \beta \frac{(\beta x)^{n-1}}{(n-1)!} \exp(-\beta x) \quad x \geq 0 \tag{5}$$

On the physical point of view the works of MASRY et.al.[23] and BEUTLER[21] showed the interest of such sampling process : alias free estimation of the signal spectra.

3. DISCRETE REPRESENTATION OF A SIGNAL

3.1. Some results

An exhaustive presentation is made by BOZZO[14], but we can briefly say that the discretization of a continuous function is a decomposition of this function according to a complete orthonormal basis of functions belonging to L^2.

So if

$f(t)$ is the function to discretisize $\in L^2$

$\{\Phi_n(t)\}$ is the complete orthonormal basis $\in L^2$

we have an isomorphism I transforming the function f into the sequence $\{f_n\}$:

$$f(t) \overset{I}{\rightarrow} \{f_n\} \in L^2_D$$

with

$$f(t) = \lim_{R \to +\infty} \sum_{n=-R}^{+R} f_n \, \Phi_n(t) \tag{6}$$

The sequence $\{f_n\}$ leads us to define the Z-transform of the function $f(t)$:

$$F(Z) = \lim_{N \to +\infty} \sum_{n=-N}^{+N} f_n \, Z^{-n} \qquad Z \in C \tag{7}$$

In our problem, $f(t)$ is the autocorrelation function of a signal $x(t)$ (possessing all the properties announced in the introduction)

If we take the LAPLACE transform of the equation (6) we have :

$$F_p(p) = \lim_{R \to +\infty} \sum_{n=-R}^{+R} f_n \, \phi_n(p) \tag{8}$$

Starting from the results showed in [17], on the conservation of the convolution product, we have

$$\phi_n(p) = \left[\phi_1(p)\right]^n \tag{9}$$

so

$$F_p(p) = \lim_{N \to +\infty} \sum_{n=-N}^{+N} f_n \left\{\left[\phi_1(p)\right]^{-1}\right\}^{-n} \tag{10}$$

Subsequently, we have

$$Z^{-1} = \phi_1(p) \tag{11}$$

This relationship shows the link between the Z-variable and the Φ_i function.

If we take the periodic case $\left(f_n = f(nT)\right)$ we have

$$Z = \exp(Tp) \rightarrow \delta(t - T)$$

$$\overset{(\text{LAPLACE})^{-1}}{}$$

In the BOZZO's paper [14], we can show the relation

$$F_p(p) = F(Z)\Big|_{Z = T(p)} \tag{12}$$

3.2. Fondamental result

We present here some remarks on the transformation

$$Z = T(p) \tag{13}$$

We know that

$$F(Z) = \sum_{-\infty}^{+\infty} f_n \, Z^{-n} \tag{14}$$

from a classical theorem, we have :

$$f_n = \frac{1}{2\pi j} \oint F(Z) \, Z^n \, \frac{dz}{Z} \tag{15}$$

taking into account (12),

$$f_n = \frac{1}{2\pi j} \int_{-j\infty}^{+j\infty} F\left(T(p)\right) \left[T(p)\right]^{n-1} \frac{\delta T}{\delta p} \, dp \tag{16}$$

LAGUERRE BASIS CASE

The transform $T(p)$ is $T(p) = \dfrac{1 + p}{1 - p}$

$$\leftrightarrow Z = \frac{1 + p}{1 - p}$$

$$\leftrightarrow p = \frac{1 - Z}{1 + Z}$$

let

$$Z = \exp(j\omega) \text{ and } p = j\Omega$$

we have

$$\Omega = \operatorname{tg} \frac{\omega}{2} \tag{17}$$

The relationship showes the main bilinear properties : alias free estimation of the signal spectra. We found here the same property we had with the spectrum estimation computed from randomly sampled signal.

Before going further in the analogy, the relation between LAGUERRE and POISSON transform can be devined, and we give here the result introduced by BOLGIANO [7]

$$f_n^{\,1} = \sum_{o}^{n} (-1)^k \, C_n^{\,k} \, f_n^{\,p} \tag{18}$$

$$f_n^{\,p} = \sum_{o}^{n} (-1)^{+k} \, C_n^{\,k} \, f_n^{\,1} \tag{19}$$

4. LINK BETWEEN RANDOM SAMPLING ACCORDING TO A POISSON LAW AND POISSON TRANSFORM

4.1. Introduction

We recall here the main ideas briefly introduced

→ both methods leads to alias-free estimation of the spectrum

→ the probability density function in the POISSON sampling case is given by

$$g_n(x) = \beta \frac{(\beta x)^{n-1}}{(n-1)!} \exp(-\beta x) \quad x \geq 0$$

→ the $f_n^{\,p}$ can easily be computed from the $f_n^{\,L}$

→ if $Z = T(p)$ and $F(Z) = \sum_{-\infty}^{+\infty} f_n \, Z^{-n}$

$$f_n = \frac{1}{2\pi j} \int_{-j\infty}^{+j\infty} F\left[T(p)\right] \left[T(p)\right]^{n-1} \frac{\partial T}{\partial p} \, dp$$

We are able now to derive the link between both approaches of the discretization problem.

4.2. Demonstration of the relationship

We know that

$$f(t) = \sum_{n=0}^{\infty} f_n^{\,p} \, P_n(t)$$

$P_n(t)$ = nth POISSON function

$$= \beta \frac{(\beta t)^{n-1}}{(n-1)!} \exp(-\beta t) \quad t \geq 0$$

so

$$f_n^{\,p} = \frac{1}{2\pi j} \int_{-j\infty}^{+j\infty} F\left[T(p)\right] \left[T(p)\right]^{n-1} \frac{\partial T}{\partial p} \, dp \tag{20}$$

$T(p)$ = {LAPLACE transform of the first POISSON function}$_{\,p \to -p}$

That is to say

$$T(p) = \frac{\beta}{\beta - p} = Z \tag{21}$$

and

$$\frac{\partial T}{\partial p} = \frac{\beta}{(\beta - p)^2} = \frac{1}{\beta} \, T^2(p)$$

all this leads to

$$f_n^P = \frac{1}{2\pi j\beta} \int_{-j\infty}^{+j\infty} F_p(p) \left[\frac{\beta}{\beta-p}\right]^{n+1} dp \qquad (22)$$

On the other hand, a process signal x(t) being done and f(t) is its correlation function. We note F(p) the bilateral LAPLACE transform of the correlation function.

$$F_p(p) = \int_{-\infty}^{+\infty} f(t) \exp(-pt)\, dt$$

We know also, that the SPP $\{t_n\}$ used to sample the signal x(t) is a POISSON distributed SPP.
So we have,

$$g_n(\tau) = \beta \frac{(\beta\tau)^{n-1}}{(n-1)!} \exp(-\beta\tau)$$

(see chapter 2)

Sampling the signal x(t) according to this SPP, leads to define the correlation coefficients as

$$f_n = E\left[x(t_m)\ \overline{x}\ (t_{m+n})\right] \qquad (23)$$

where

$$E[.] \triangleq E_x\left[E_t(.)\right] = E_t\left[E_x(.)\right] \qquad (24)$$

That is to say : E(.) acts as a double averaging operator on both signal and time. Of course processes x(t) and $\{t_i\}$ being independant, the order of the averages doesn't matter.

$$f_n = E_t\left[E_x\left[x(t_m)\right] \cdot \overline{x}\ (t_{m+n})\right] \qquad (25)$$

$E_x(.)$ = averaging according x(t) process

$E_t(.)$ = averaging according $\{t_i\}$ process

The autocorrelation function of a stationary signal x(t) is

$$f(\tau) = E_x\left[x(t)\ \overline{x}\ (t+\tau)\right]$$

with $t_m = t_n$ $\quad t_{m+n} = \tau + t$

$$f(\tau) = E_x\left[x(t_m)\ \overline{x}\ (t_{m+n})\right]$$

We have here :

$$\begin{cases} f(\tau) = \dfrac{1}{2\pi j} \displaystyle\int_{-j\infty}^{+j\infty} F_p(p) \exp\left[p(t_{m+n} - t_m)\right] dp \\ \tau = t_{m+n} - t_n \\ F_p(p) = \text{Bilateral LAPLACE transform of the autocorrelation function} \end{cases} \qquad (26)$$

This leads to

$$E_x\left[x(t_m)\ \overline{x}\ (t_{m+n})\right] = \frac{1}{2\pi j} \int_{-j\infty}^{+j\infty} F_p(p) \exp\left(p(t_{m+n} - t_n)\right) dp$$

and finaly

$$f_n = E_t \left[\frac{1}{2\pi j} \int_{-j\infty}^{+j\infty} F_p(p) \exp\left(p(t_{m+n} - t_m)\right) dp \right]$$

$$f_n = \frac{1}{2\pi j} \int_{-j\infty}^{+j\infty} F_p(p) \; E_t \left[\exp\left(p(t_{m+n} - t_m)\right)\right] dp \qquad (27)$$

$\{t_n\}$ is a SPP and the random variable $L_n(t)$ (see chapter 2) is very interesting because, if we let

$$t = t_m$$

$$L_n(t) = L_n(t_m) = t_{m+n} - t_n = L_n \text{ (stationary property)}$$

and we have

$$E_t \left[\exp\left(p(t_{m+n} - t_n)\right)\right] = E_t \left[\exp(pL_n)\right]$$

$$E_t \left[\exp(pL_n)\right] = \int_{-\infty}^{+\infty} g_n(t) \exp(pt)\, dt = \hat{g}_n(-p) = \left[\frac{\beta}{\beta - p}\right]^n \qquad (28)$$

Which is the definition of the generating function of the process $L_n(t)$

So we have

$$f_n = \frac{1}{2\pi j} \underbrace{\int_{-j\infty}^{+j\infty} F_p(p) \left[\frac{\beta}{\beta - p}\right]^n dp}_{f^p_{n-1}} \qquad (29)$$

The relationship is then obvious :

$$\boxed{f_n = \beta f^p_{n-1}} \qquad (30)$$

Périodical case :

In this case we have :

$$t_n = (n-1)T$$

so

$$P(L_n < t) = 0 \qquad t<(n-1)T$$

$$P(L_n < t) = 1 \qquad t\geq(n-1)T$$

and

$$G_n(t) = H\left(t-(n-1)T\right) \qquad (31)$$

$$= \text{one step function at } t = (n-1)T$$

The probability density function of the process generating the sampling instants is (the derivative are taken in the generalized function sense)

$$g_n(t) = G'_n(t) = \{G_n(t)\}' + \delta\left(t-(n-1)T\right)$$

$$g_n(t) = \delta\left(t- (n-1)\, T\right) \qquad (32)$$

The generating function is

$$\int_{-\infty}^{+\infty} \delta\left[t-(n-1)T\right] \quad \exp(pt) \quad dt = \exp\left[(n-1)Tp\right]$$

then

$$T(p) \quad = \quad \hat{g}_n(-p) \quad = \left[\exp(Tp)\right]^{n-1}$$

and

$$f_n \quad = \quad \frac{1}{2\pi j} \quad \int_{-j\infty}^{+j\infty} \quad F_p(P) \quad \left[\exp(Tp)\right]^{n-1} \; dp \qquad (33)$$

Classical sampling gives :

$$f(nT) \quad = \quad \frac{1}{2\pi j} \quad \oint F(Z) \; Z^n \quad \frac{dz}{z} \qquad (34)$$

with $Z = \exp(Tp)$

We know that :

$$\begin{cases} F(Z) \quad = \quad F_p(p) \\ f_n \quad = \quad \frac{1}{2\pi j} \quad \int_{-j\infty}^{+j\infty} F_p(P) \; Z^{n-1} \; dp \\ dZ \quad = \quad TZ \; dp \end{cases}$$

This leads to

$$f_n \quad = \frac{1}{2\pi j} \quad \oint \quad F(Z) \quad Z^{n-1} \quad \frac{dz}{z}$$

and finaly :

$$\boxed{f_n \quad = \quad \frac{1}{T} \quad f\left[(n-1)T\right]} \qquad (35)$$

to reach this well known result, we take the same approach as the one presented in the first part of the chapter. A total analogy can be established if we define the parameter β as an average samply rate.

5. CONCLUSION

What we intend to show in this paper is a strong consistency in the two main discretization ways (in the random signal field)

- → discretization using a numerical transform like LAGUERRE and POISSON transform (linked with bilinear transform)
- → sampling technics in the most general case : random sampling

This consistency has been shown as an exemple : sampling according a POISSON law and POISSON transform.

In fact, we can say that the choice of the function basis on which we want to discretisize the correlation function must help us to choice the law according to which, we want to sample the signal.

What it should be interesting now is to developp a generalization of such approach and to point out how the generating function act as a bridge between the two ways.

BIBLIOGRAPHY

[1] N. WIENER "Generalized Harmonic Analysis" Acta Mathematica - Volume 55 pages 117-258 1930

[2] Y.W. LEE "Synthesis of Electrical Network by Means of the FOURIER Transforms of LAGUERRE Functions" J. Math. Physi. 11 - 83/113 - 1931

[3] K. STEIGLITZ "The General Theory of Digital Filters with Applications to Spectral Analysis" - AFOSR Report, n° 64-1664 (Eng. Sc. D. Dissertation NEW YORK University - NEW YORK) - 1963

[4] K. STEIGLITZ "The Approximation Problem of Digital Filters" - Tech. Report, n° 400-56 (Dep. Of Electrical Engineering, NEW YORK University, NEW YORK 1963

[5] K. STEIGLITZ "Rational Transform Approximation via the LAGUERRE Spectrum" - Journal of the Franklin Institute, Volume 280, number 5 Novembre 1965

[6] K. STEIGLITZ "The Equivalence of Digital and Analog Signal Processing" - Information and Control 8, pages 445 à 467 - 1965

[7] L.P. BOLGIANO and M.J. PIOVOSO "POISSON Transform Signal Analysis" IEEE Transactions on Information Theory - Vol. IT-14 pp 600-601 July 1968

[8] M.J. PIOVOSO and L.P. BOLGIANO "Digital Simulation using POISSON Transform Sequences" dans "Computer Processing in Communications" édité par J. FOX MRI Symposia Series Vol. 19, pp 505-514 - Polytechnic Press BROOKLYN - 1970

[9] L.P. BOLGIANO and L.T. QUICK "Obtaining Digital Signals for POISSON Filtering from Periodic Samples" Proc. IEEE Vol. 63 (September) pp 1366-1367 1976

[10] L.T. QUICK and L.P. BOLGIANO "Deconvolution by POISSON Transformation" Conference Record IEEE International Conference on Acoustics, Speech and Signal Processing, PHILADELPHIA, Pa - pp 350-353 1976

[11] A.V. OPPENHEIM and D.M.JOHNSON "Discrete Representation of Signals" Proceeding of the IEEE - Vol. 60 n° 6 (June) - 1972

[12] C. BOZZO "Identification expérimentale des processus Analyseur de Spectre de LAGUERRE" Etude A 20 GESTA CAPCA du 14 Février 1969 (DCAN DE TOULON)

[13] C. BOZZO "Etude d'un isomorphisme entre l'espace des signaux continus et l'espace des signaux discrets. Différentes applications" Thèse de Doctorat es Sciences Marseille - 1970

[14] C. BOZZO "Notion de transformée de LAGUERRE d'un signal continu" RAIRO J1 - 1972

[15] C. BOZZO "Transformée de LAGUERRE d'un signal continu : relation entre les transformées en ζ et en Z" CRAS t 277 p. 189-192 - 1973

[16] C. BOZZO "Transformée de LAGUERRE d'un signal continu - Application à l'étude du régime asymptotique du filtre de KALMAN" - CRAS t 277 p. 225-228 - 1973

[17] C. BOZZO " Modélisation de signaux continus et caractérisations discrètes - Transformation de LAGUERRE " Colloque GRETSI 26/30 Avril 1977 (Nice FRANCE)

[18] M. ROESCH, J. RAGOT, P. HEITZMANN "Identification d'un système linéaire sous forme d'une série de fonctions de LAGUERRE à l'aide de séquences binaires - Revue A Volume 17 n° 1 - BRUXELLES 1976

[19] C. BORGET et P. FAURE "Algorithmes de factorisation approchée utilisant les fonctions orthogonales" RAIRO J - 1973

[20] F.J. BEUTLER and OAZ LENEMAN "The Theory of Stationary Point Process" Acta Mathématica 116 pp 160-197
"Random sampling of random processes - Stationary Point Process Information and Control 9 p. 325-346 - 1966

[21] F.J. BEUTLER "Alias Free Randomly Timed Sampling of stochastic Processes" - IEEE trans. on Information Theory
Vol. IT - 16 N° 2 - 1970

[22] E. MASRY and M.M.C. LUI "Discrete time spectral estimation of continuous parameters processes - A new consistent estimate"
IEEE trans. on Information Theory - Vol. IT-32 n°3 - 1976

[23] E. MASRY, D. KRAMER and C. MIRABILE "Spectral Estimation of continuous time process : Performance comparaison between Periodic and POISSON sampling schemas".
IEEE Trans. on Automatic Control - Vol. AC.23 - n°4 - 1978

[24] A.J. JERRY "The Shannon Sampling Theorem - It's Various Extensive and Applications : A theorical review"
Proceedings of IEEE Vol. n°4 - 1977

A ROBUST CONJUGATE DIRECTIONS METHOD FOR SOLVING LINEAR SYSTEMS

BARRY J. SULLIVAN
Dept. of Elec. Engineering
and Computer Science
Northwestern University
Evanston, Illinois 60201

BEDE LIU
Dept. of Elec. Engineering
and Computer Science
Princeton University
Princeton, New Jersey 08544

ABSTRACT

The method of conjugate gradients is the most popular of several conjugate directions methods. In this paper, a second conjugate directions method, known as A-minimal iterations, is examined. It differs from conjugate gradients primarily in that the direction vectors are computed in a manner independent of the observations. When the observations are the dominant noise source in the problem, this difference makes A-minimal iterations the more robust of the two methods. Examples from the area of signal extrapolation support this assertion.

INTRODUCTION

Researchers in many areas have been attracted to the conjugate gradient method for solving systems of linear equations. Image reconstruction [1] and signal extrapolation [2] are just two of these areas. The attractions of the method are its ease of implementation relative to eigenanalysis methods and its fast convergence relative to other descent methods [3]. A major drawback of conjugate gradients is the difficulty of analyzing error effects, including the effect of noise-corrupted data.

Like conjugate gradients, the method of A-minimal iterations is a conjugate directions method. The important difference between the two is the manner in which the direction vectors are computed. In the conjugate gradient method, the direction vectors are computed using the observation vector, leading to the difficulties in error analysis noted above. The direction vectors for the method of A-minimal iterations are computed in a manner independent of the observations. Not only does this simplify the analysis of the effect of noise entering through the observations, it also makes the method of A-minimal iterations more robust than conjugate gradients in the presence of such noise.

The method of A-minimal iterations and a robust computational procedure are presented first. An analysis of the improvement of the solution from one iteration to the next leads to a criterion for identifying direction vectors that are closer to the signal than the noise. A computational procedure based on this criterion is proposed. To illustrate the results and support the assertion that this method is a more robust conjugate directions method than conjugage gradients, examples of signals extrapolated using A-minimal iterations are presented.

METHOD OF A-MINIMAL ITERATIONS

The method of a A-minimal iterations can be applied to any system of linear equations, with appropriate modifications [4]. Take as the basic system

$$Ax = y$$

where A is (m x n) and real. The vectors x and y, appropriately dimensioned, are also real. The minimum-norm least-squares (MNLS) solution of this system is

$$x^+ = A'(AA')^{-1}y \ , \tag{1}$$

with prime (') denoting transposition. To simplify notation, define a matrix C as the product AA' and rewrite (1) in the form

$$x^+ = A'z \ ,$$

where z is equal to $C^{-1}y$. The problem then becomes one of solving the system

$$Cz = y \tag{2}$$

for z and applying A' to obtain x^+. If the matrix C is positive definite, the method of A-minimal iterations can be applied to solve equation (2) for z [4].

The solution to (2) by A-minimal iterations is given by

$$z = z_0 + \sum_{i=0}^{m-1} \alpha_i q_i \tag{3}$$

where $\{q_i\}_{i=0}^{m-1}$ is a set of m-vectors, linearly independent and pairwise conjugate with respect to C [4]. Using the conjugate property of the direction vectors, $q_i'Cq_j=0$ for $i\neq j$, the parameters α_i can be determined as

$$\alpha_i = \frac{y'q_i}{q_i'Cq_i} \ .$$

The conjugate vectors are computed according to

$$q_1 = Cq_0 - \gamma_0 q_0 \tag{4}$$

$$q_{i+1} = Cq_i - \gamma_i q_i - \delta_i q_{i-1} \qquad (1 \leq i \leq m-2)$$

with $q_0\neq 0$ chosen arbitrarily. The multipliers γ_i and δ_i are determined by the expressions

$$\gamma_i = \frac{q_i'C^2q_i}{q_i'Cq_i} = \frac{q_i'C^2q_i}{q_iC^2q_{i-1}} \ , \qquad \delta_i = \frac{q_i'C^2q_{i-1}}{q_{i-1}'Cq_{i-1}} = \frac{q_i'Cq_i}{q_{i-1}'Cq_{i-1}} \ .$$

The solution can also be computed iteratively as

$$z_{k+1} = z_k + \alpha_k q_k \ . \tag{5}$$

Using the residual at each step, $r_k=y-Cz_k$, an alternate formula for α_k is the equation

$$\alpha_k = \frac{r_k'q_k}{q_k'Cq_k} \ . \tag{6}$$

The computational procedure is then to choose $q_0\neq 0$ and generate $\{q_i\}_{i=1}^{m-1}$ according to (4), using the expressions

$$\gamma_i = \frac{q_i'C^2q_i}{q_i'Cq_i} \ , \quad \delta_i = \frac{q_i'Cq_i}{q_{i-1}'Cq_{i-1}}$$

to minimize storage and computation. With the standard choice of $z_0=0$, the iteration given by equations (5) and (6) is carried out until z_m is reached or some stopping criterion is met. Calling this vector $\hat{z}$, the computed solution to the original system is then $\hat{x}=A'\hat{z}$.

One reason for stopping the iteration before z_m is reached is a premature breakdown in the generation of conjugate vectors. Breakdown is said to occur at the r-th step if $q_r=0$ or the product $q_r'Cq_r$ is effectively zero to the precision of machine arithmetic. Premature breakdown -- breakdown occurring for $r < m$ -- may be the result of a poor choice of q_0 or poor condition of the problem. In either case, the set of conjugate vectors can be completed by determining q_r according to

$$q_r = w - \sum_{i=0}^{r-1} \beta_i q_i \ , \qquad \beta_i = \frac{w'Cq_i}{q_i'Cq_i}$$

where w is an arbitrary non-zero m-vector. The remaining vectors are generated according to (4) with q_r as the initial vector until the set is completed or the next breakdown occurs.

The assertion that the method of A-minimal iterations as just described is more robust than conjugate gradients is based on the procedures for generating the direction vectors, which is where the two methods differ. For both methods, the direction vectors are conjugate with respect to the matrix of the system being solved. In the conjugate gradients method, the calculation of the direction vectors at each step proceeds from the solution at the previous step [3] and is thus dependent on the observations (the vector y as formulated here). In contrast, the entire set of direction vectors used in A-minimal iterations is computed in a manner independent of the observations. If it is assumed that the dominant source of error is noise entering through the observations, the method of A-minimal iterations should be the more robust of the two, as is reflected in the examples of signal extrapolation below.

NOISE ANALYSIS

The noise for this analysis is assumed to enter solely through the vector of observations, the matrix C being relatively noise-free. Suppose then, that instead of y, $\tilde{y}=y+y_e$ is known, where y_e is the vecor of observation errors. Unlike the direction vectors for the method of conjugate gradients, the q_i can be assumed to be computed without error. The solution at each iteration given the observations $\tilde{y}$ is written as

$$\tilde{z}_{k+1} = \tilde{z}_k + \tilde{\alpha}_k q_k \qquad (7)$$

with the parameters $\tilde{\alpha}_k$ determined from the equation

$$\tilde{\alpha}_k = \frac{\tilde{r}_k'q_k}{q_k'Cq_k} \quad (\tilde{r}_k=\tilde{y}-C\tilde{z}_k) \ .$$

Mean-square error

One type of error of interest is the error between the desired solution at each iteration and the computed one. It is appropriate to study this error in both the C-metric, as the method of A-minimal iterations progresses toward the solution in this metric [4], and the

Euclidean metric, as the application here is finding the MNLS solution of a system of equations. Define the error vector e_k as the difference between these vectors, $z_k - \tilde{z}_k$. This error vector can also be written recursively as

$$e_k = e_{k-1} + \xi_{k-1}\, q_{k-1} = \sum_{i=0}^{k-1} \xi_i q_i \qquad (8)$$

where the conjugate property of the direction vectors has been utilized and the parameter ξ_i is given by

$$\xi_i = \frac{y_e' q_i}{q_i' C q_i} .$$

The length of this vector in the metric defined with respect to C [5] is found to be

$$\|e_k\|_C = \left[\sum_{i=0}^{k-1} \xi_i^2 q_i' C q_i \right]^{1/2} . \qquad (9)$$

In the Euclidean metric, its length is

$$\|e_k\|_2 = \left[\sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \xi_i \xi_j q_i' q_j \right]^{1/2} . \qquad (10)$$

The mean-square value of e_k can be determined in each of the metrics. From equation (9) and the expression for ξ_i given with (8),

$$E(\|e_k\|_C^2) = \sum_{i=0}^{k-1} \frac{q_i' R_e q_i}{q_i' C q_i} , \qquad (11)$$

where R_e is the (m x m) autocorrelation matrix associated with the vector y_e. Similarly, the mean-square value of e_k can be found using equation (10):

$$E(\|e_k\|_2^2) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \frac{(q_i' R_e q_j)(q_i' q_j)}{(q_i' C q_i)(q_j' C q_j)} . \qquad (12)$$

From the positive definite nature of R_e and C, it is immediately apparent that equation (11), the mean-square error in the C-metric, is increasing with k. If y_e is taken to be white noise with a mean-square value of σ^2, equation (12) becomes

$$E(\|e_k\|_2^2) = \sigma^2 \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \frac{(q_i' q_j)^2}{(q_i' C q_i)(q_j' C q_j)} ,$$

and the growth of the error as the iteration progresses is evident in the Eucliean metric as well.

Expressions for the mean square error between the desired final solution, $z = z_m$, and the computed solution at the k-th iteration, $\tilde{z}_k$, can also be derived for both metrics. In either metric, the squared norm of this error can be written as

$$\|z_m - \tilde{z}_k\|^2 = \|(z_m - z_k) + e_k\|^2 .$$

Recognizing that the difference between z_m and z_k is

$$z_m - z_k = \sum_{i=k}^{m-1} \alpha_i q_i ,$$

the expression for the squared norm becomes

$$\|z_m - \tilde{z}_k\|_2^2 = \sum_{i=k}^{m-1} \sum_{j=k}^{m-1} \alpha_i \alpha_j q_i' q_j + 2 \sum_{i=k}^{m-1} \sum_{j=0}^{k-1} \alpha_i \xi_i q_i' q_j + \|e_k\|_2^2$$

in the Euclidean metric, and

$$\|z_m - \tilde{z}_k\|_C^2 = \sum_{i=k}^{m-1} \alpha_i^2 q_i' C q_i + 2 \sum_{i=k}^{m-1} \alpha_i \xi_i q_i' C q_i + \|e_k\|_C^2$$

in the C-metric.

Assuming that the signal and the noise are uncorrelated and denoting the signal autocorrelation matrix as R_y, an expression for the mean-square error in the Euclidean metric can now be written as

$$E(\|z - \tilde{z}_k\|_2^2) = \sum_{i=k}^{m-1} \sum_{j=k}^{m-1} \frac{(q_i' R_y q_j)(q_i' q_j)}{(q_i' C q_i)(q_j' C q_j)} + E(\|e_k\|_2^2) , \qquad (13)$$

with $E(\|e_k\|^2)$ as given in (12). In the case of band-limited signal extrapolation, R_y is equal to C, leading to the simplified expression

$$E(\|z - \tilde{z}_k\|_2^2) = \frac{q_i' q_i}{q_i' C q_i} + E(\|e_k\|_2^2) .$$

If the noise is assumed to be white, this expression is easily separated into a part increasing in k and a part decreasing in k.

An expression for the mean-square error in the C-metric can be written as

$$E(\|z - \tilde{z}_k\|_C^2) = \sum_{i=k}^{m-1} \frac{q_i' R_y q_i}{q_i' C q_i} + E(\|e_k\|_C^2) , \qquad (14)$$

with $E(\|e_k\|_C^2)$ as given in (11). The division of this expression into a part increasing in k and a part decreasing in k is readily apparent. By taking R_y equal to C as in the extrapolation problem, and y_e as white noise with mean-square value σ^2, equation (14) simplifies to

$$E(\|z - \tilde{z}_k\|_C^2) = (m - k) + \sigma^2 \sum_{i=0}^{k-1} \frac{q_i' q_i}{q_i' C q_i} .$$

Improvement of solution

It is clear from the preceding analysis that for a sufficiently high noise level, the growth of the mean-square value of e_k can overtake the

decrease in the overall mean-square error as the iteration progresses. For this reason, it may not be desirable to make use of all the available direction vectors. A criterion for selecting those direction vectors most likely to have the strongest signal components is developed here, based on a consideration of the improvement of the computed solution in successive iterations.

The error function $\tilde{\varepsilon}_k=(z-\tilde{z}_k)'C(z-\tilde{z}_k)$ is used to study the effect of the noise on the improvement of the solution from one iteration to the next. The change in the error function as it is evaluated at successive iterations is

$$\begin{aligned}\tilde{\varepsilon}_k - \tilde{\varepsilon}_{k+1} &= (\tilde{z}_k'C\tilde{z}_k - \tilde{z}_{k+1}'C\tilde{z}_{k+1}) - 2(y'\tilde{z}_k - y'\tilde{z}_{k+1}) \\ &= -\tilde{\alpha}_k^2 q_k'Cq_k + 2\tilde{\alpha}_k y'q_k \ .\end{aligned}$$

Substituting the expression for $\tilde{\alpha}_k$ given with equation (7) leads to

$$\tilde{\varepsilon}_k - \tilde{\varepsilon}_{k+1} = \frac{(y'q_k)^2 - (y_e'q_k)^2}{q_k'Cq_k} \ .$$

The criterion to be satisfied at each iteration to guarantee improvement over the solution already obtained is then

$$\Delta_k \triangleq (y'q_k)^2 - (y_e'q_k)^2 > 0 \ . \tag{15}$$

The direction vectors can be divided into two sets, Q and Q_e, based on the criterion (15):

$$Q = \{q_i: \Delta_k > 0\}, \quad Q_e = \{q_i: \Delta_k \leq 0\} \ .$$

Recalling the definition of the angle between two vectors [5], let

$$\theta(q_i) = \cos^{-1}\left[\frac{y'q_i}{(y'y)^{1/2}(q_i'q_i)^{1/2}}\right] \ ,$$

$$\theta_e(q_i) = \cos^{-1}\left[\frac{y_e'q_i}{(y_e'y_e)^{1/2}(q_i'q_i)^{1/2}}\right] \ .$$

It is clear that $\theta(q_i) > \theta_e(q_i)$ for all $q_i \in Q$; that is, the vectors in Q can be considered closer to the signal part of $\tilde{y}$ than the noise. Thus, confining the iteration to those directions guaranteeing an improvement in the error function at each step is equivalent to basing the solution on those direction vectors that are closer to the signal than the noise.

The unfortunate aspect of the above discussion is that Q cannot be distinguised from Q_e in practice, as exact knowledge of y and y_e is required. However, if y and y_e are treated as random vectors, a usable criterion can be derived from (15). Let R_y and R_e again be the (m x m) autocorrelation matrices associated with y and y_e, respectively. Assuming for simplicity that both y and y_e are zero mean, the expected value of Δ_k is then

$$E(\Delta_k) = q_k'R_yq_k - q_k'R_eq_k \ , \tag{16}$$

and the criterion for including q_k in Q is that $E(\Delta_k)$ be greater than zero. For the extrapolation problem, R_y is the matrix C. If y_e represents samples from a white noise process with mean-square value σ^2, R_e is equal to $\sigma^2 I$. In this case, (16) becomes

$$E(\Delta_k) = q_k' C q_k - \sigma^2 q_k' q_k$$

and the criterion is that the length of q_k in the C-metric be greater than its Euclidean length weighted by the mean-square value of the noise. The utility of this criterion for selecting direction vectors in determining a stopping point for the iteration is demonstrated by the extrapolation examples.

EXAMPLES

The discussions of this paper are illustrated by applying the method of A-minimal iterations to the problem of extrapolating a band-limited signal in discrete-time. Extrapolation can be viewed as solving an underdetermined system of linear equations [2], [6]. Take y to be the m-vector of observations and x to be an n-vector representing the extrapolated sequence. The two vectors are related throught the (m x n) truncation operator T with elements

$$t_{ij} = \begin{cases} 1 & i=j, \quad |i| \leq N \qquad (i=0,\pm,\ldots,\pm N) \\ 0 & \text{otherwise}, \qquad (j=0,\pm 1,\ldots,\pm M) \end{cases}$$

and the (n x n) band-limiting operator B with elements

$$b_{ij} = \frac{\sin 2\pi f_2(i-j)}{\pi(i-j)} - \frac{\sin 2\pi f_1(i-j)}{\pi(i-j)},$$

as y=TBx. Here, m is equal to 2M+1, n is equal to 2N+1, and M is less than or equal to N. Identifying A as the composite of the operators T and B, the MNLS extrapolation of y can be expressed as in equation (1). The (m x m) symmetric matirx C=TBT' has been shown to be positive definite [2], so the method of A-minimal interations can be applied to solve Cz=y, obtaining $\hat{z}$. The extrapolation $\hat{x}$ is then found by applying BT' to $\hat{z}$.

The extrapolation of a 45-point sequence to a 195-point sequence is the basis for all the examples. The original 195-point sequence is generated by band-limiting a random sequence to the desired bandwidth. Noise is added to the 45-point sequence taken from the center of the original sequence and the extrapolation is carried out. The result of using A-mininal iterations to extrapolate such a sequence is illustrated by Figure 1. The figure shows the extrapolation after six iterations at a signal-to-noise ratio of 30dB along with the true 195-point sequence. The normalized mean-square error in the Euclidean metric between the true and extrapolated sequences is 0.4295.

The effect of using the expected value of Δ_k, equation (16), to select the direction vectors for this example is illustrated by Figure 2. The same observation interval used in generating Figure 1 is used again here, also at a signal-to-noise ratio of 30dB. After the first five iterations, subsequent direction vectors are rejected for failing the selection criterion. The result (Fig. 2) can be seen to be quite similar to the optimal solution by A-minimal iterations (Fig. 1). The normalized

mean-square error between this extrapolation and the true sequence is 0.4397, again quite close to that seen in the optimal case.

The performance of the method of conjugate gradients at this signal-to noise level is illustrated by Figure 3. The superiority of A-minimal iterations is evident by comparing this figure to Figures 1 and 2, and also by comparing Tables I and II. Table I gives the normalized mean-square error measured at various noise levels with A-minimal iterations, while Table II presents similar data for conjugate gradients. As expected, the measured mean-square error is less for A-minimal iterations at higher noise levels. The performance of the two methods becomes comparable as the noise decreases.

TABLE I
Measured Error for Extrapolation by A-Minimal Iterations

SNR	Error	SNR	Error	SNR	Error
3.0	1.0878	18.0	0.6576	33.0	0.4305
6.0	0.8889	21.0	0.5882	36.0	0.4271
9.0	0.7720	24.0	0.5050	39.0	0.4263
12.0	0.7077	27.0	0.4609	42.0	0.4265
15.0	0.6742	30.0	0.4397	45.0	0.4271

TABLE II
Measured Error for Extrapolation by Conjugate Gradients

SNR	Error	SNR	Error	SNR	Error
3.0	1.8671	18.0	0.6590	33.0	0.6210
6.0	1.0837	21.0	0.6499	36.0	0.6156
9.0	0.8453	24.0	0.6454	39.0	0.4634
12.0	0.7130	27.0	0.6435	42.0	0.4159
15.0	0.6761	30.0	0.6429	45.0	0.4144

REFERENCES

[1] T.F. Budinger, G.T. Gullberg and R.H. Huesman, "Emission computed tomography," in Image Reconstruction from Projections, G.T. Herman, ed., New York: Springer-Verlag, 1979.

[2] A.K. Jain and S. Ranganath, "Extrapolation algorithms for discrete signals with application in spectral analysis," IEEE Trans. on Acoust., Speech, and Signal Proc., vol. ASSP-31, no. 1, pp. 830-845, Aug. 1981.

[3] H.R. Schwarz, H. Rutishauser and E. Stiefel, Numerical Analysis of Symmetric Matrices, P. Hertelendy, trans., Englewood Cliffs, NJ: Prentice-Hall, 1973.

[4] D.K. Faddeev and V.N. Faddeeva, Computational Methods of Linear Algebra, R.C. Williams, trans., San Francisco: W.H. Freeman & Co., 1963.

[5] M.C. Pease, Methods of Matrix Algebra, New York: Academic Press, 1965.

[6] B.J. Sullivan and Bede Liu, "On the use of singular value decomposition and decimation in discrete-time band-limited signal extrapolation," IEEE Trans. on Acoust., Speech, and Signal Proc., Dec., 1984.

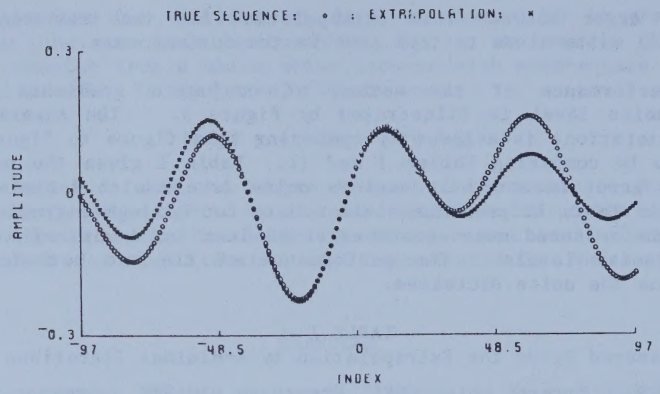


Fig. 1. Extrapolation using A-minimal iterations (six iterations).

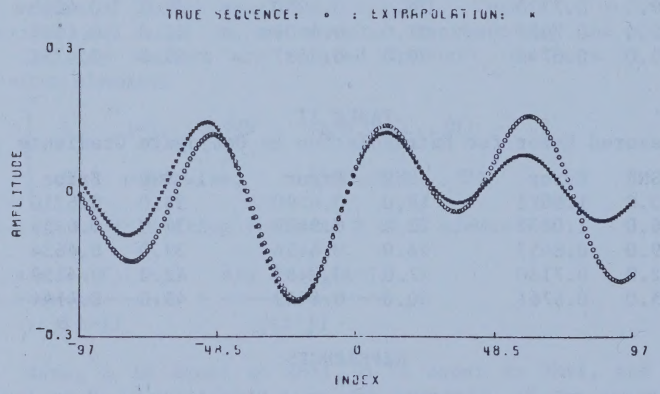


Fig. 2. Extrapolation using A-minimal iterations with selection of direction vectors (five iterations).

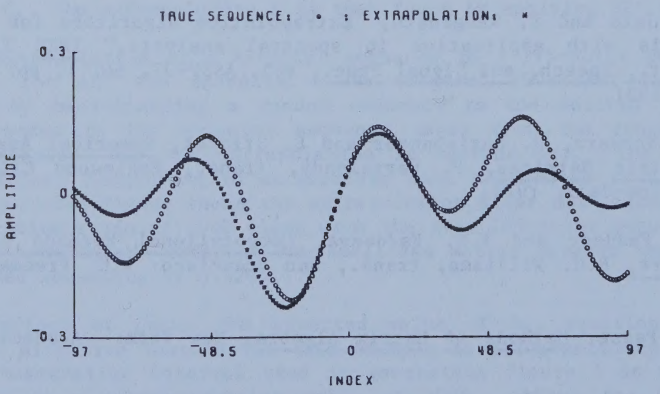


Fig. 3. Extrapolation using conjugate gradients.

OPTIMUM AMPLITUDE WEIGHTING FOR RANGE SIDELOBE SUPPRESSION IN A HIGH RESOLUTION RADAR HYBRID WAVEFORM DESIGN

DANIEL B. KOCH
Emerson Electric Company
St. Louis, Missouri

WILLIAM H. TRANTER
University of Missouri - Rolla
Rolla, Missouri

ABSTRACT

A "hybrid" waveform design technique, utilizing both complementary phase coding and linear frequency stepping, is presented for achieving improved slant range resolution in a synthetic aperture radar. The "composite" ambiguity function, defined to be the coherent sum of the individual pulse ambiguity functions, is used to describe the matched filter output for the hybrid waveform analytically. A method for selecting optimum amplitude weights, based on a cost function related to the integrated sidelobe ratio of the compressed waveform, is also presented. This is contrasted with the usual method of choosing a window function with a low peak sidelobe ratio. The results of a computer study are summarized which characterize the performance of this method. Using an equivalent matrix formulation, a numerical example involving two amplitude weights is presented which can be solved in closed form. The cost function is plotted as a surface in three dimensions and illustrates why computer solutions are necessary for the general case.

I. INTRODUCTION

In this paper the results of a radar waveform design technique for improving slant range resolution in a synthetic aperture radar (SAR), in which the processor is assumed to be bandwidth limited, are presented. Rather than take the usual approach of using either phase or frequency modulation alone, a "hybrid" approach consisting of both modulation schemes is utilized. Phase coding is used to achieve a range resolution value consistent with the processing bandwidth of the SAR system. Frequency stepping is then added to further increase the range resolution.

The basic waveform under consideration, illustrated in Figure 1, is a coherent pulse train. This waveform consists of K repetitions of L pulses. These L pulses are biphase modulated by a complementary code set sequence. At this point the waveform's range resolution is consistent with the phase code chip time T_c. Next, the carrier frequency is linearly increased from repetition to repetition in discrete steps $F = 1/T_c$. This has the effect of further increasing the waveform's frequency extent and thus its range resolution. We term this scheme frequency stepped, phase coded (FSPC) modulation. One added restriction imposed by the linear frequency stepping is that the beginning phase of each pulse must be equal. This condition is required in order to achieve the desired increase in resolution.

One reason for choosing complementary phase coding and linear frequency stepping is the fact that each modulation scheme requires common processing. This processing consists of matched filtering the radar returns on a pulse-by-pulse basis and then coherently adding the results. The effect of these operations is to pulse compress the original transmitted waveform. This pulse compression occurs for two reasons. First, the use of complementary phase coding compresses the wide real pulse width to a narrow effective pulse width equal to the phase code chip time. Second, the linear frequency stepping further compresses the pulse width to a final value consistent with the total frequency stepping range. The key to achieving this result is the pulse-by-pulse matched filtering and coherent summation combination. A matched filter which is matched to the entire waveform as a whole, which is the usual case considered, would not produce this effect.

The output of the processor described above is a finite duration pulse-like waveform. Range resolution is determined by the mainlobe width of this pulse. In addition to the mainlobe, several sidelobes are present. These sidelobes are undesirable since they offer potential interference from adjacent radar targets. A common method used to reduce these sidelobes is to multiply the radar returns by a suitable window function with a low peak sidelobe ratio (PSLR) [1]. The approach taken in this design is different. Initially, a cost function is identified which is related to the integrated sidelobe ratio (ISLR) of the compressed waveform. An amplitude weight is then chosen for each of the received radar returns so as to minimize this cost function.

Closely related to the concept of matched filtering radar returns is the use of the ambiguity function to describe waveform properties [1, 2]. A plot of the ambiguity function, termed the ambiguity diagram, is drawn as a surface in three dimensions. Most authors use the ambiguity function computed for the waveform as a whole in their analyses. In our case this approach would be of little value since it does not model the processing needed for the FSPC waveform. Instead we will compute the ambiguity function on a pulse-by-pulse basis and then coherently add the results. The "composite" ambiguity function is a term which has been suggested in recent literature to describe this quantity [3].

II. HYBRID WAVEFORM COMPOSITE AMBIGUITY FUNCTION

The ambiguity function for a complex signal $s(t)$ is given by

$$X(t,f_d) = \int_{-\infty}^{\infty} s(x)\, s^*(x+t)\, \exp[j2\pi f_d x]\, dx \qquad (1)$$

In (1) a positive value for delay, t, corresponds to a target beyond a specific reference range while a positive value for doppler shift, f_d, indicates an approaching target.

The application of the ambiguity function (1) to the hybrid waveform as a whole is inappropriate since it does not model the required processing. In order to model the matched filter and coherent summer processor we must instead use the composite ambiguity function defined as

$$X'(t,f_d) = (1/N) \sum_{n=0}^{N-1} X_n(t,f_d) \qquad (2)$$

where the prime is used to distinguish this from (1). The function $X_n(t,f_d)$ is the result of applying (1) to each individual pulse in the pulse train, where n denotes the n-th pulse.

The type of phase coding which will be used is based on a complementary code set. An L-member complementary code set consists of L code sequences whose individual autocorrelation functions, when added together, reinforce for zero lag and completely cancel for other lags [4, 5]. Each member of the code set phase codes a separate pulse. There are K repetitions of this code set, each transmitted at a separate frequency, so that KL=N. In this case, (2) becomes

$$X'(t,f_d) = \{(1/L) \sum_{l=0}^{L-1} Y_l(t,f_d)\} \{(1/K) \sum_{k=0}^{K-1} \exp[-j2\pi f_k t]\} \qquad (3)$$

where $Y_l(t,f_d)$ is the ambiguity function (1) of each individual phase coded pulse disregarding

frequency modulation and where f_k denotes the k-th frequency. Note that (3) is composed of two factors. The first factor is due to phase coding and the second factor is due to frequency stepping. We see that the effects of phase coding and frequency stepping are separable when the individual ambiguity functions of each pulse are computed and then coherently added to yield the composite ambiguity function.

We next wish to modify (3) by the addition of amplitude weights and to evaluate it for zero doppler in order to model the matched filter output. By constaining the phase coding to be complemenatry and the frequency stepping to be linear with $f_k=kF/J$, the magnitude of (3) for $f_d=0$ becomes

$$|X'(t,0)| = (1/K)\ (1-|t|/T_c)\ \left|\sum_{k=0}^{K-1} a_k \exp[-j2\pi kFt/J]\right| \qquad (4)$$

where J is an integer used to parametize the fundamental step size F and where a_k represents the amplitude weighting function which is applied to all of the L pulses in a given repetition at a specific frequency f_k. Figure 2 illustrates (4), normalized in delay by T_c, for J=1 and K=4 where the a_k's have been determined by the method described in the next section.

III. OPTIMUM AMPLITUDE WEIGHTING

Inspection of (4) and Figure 2 reveals that the effect of complementary phase coding is to force the composite ambiguity function to zero along the delay axis outside the range $[-T_c, T_c]$. This effectively defines the resolution properties of the phase coding. The technique of frequency stepping introduces nulls inside the phase coding window, $1-|t|/T_c$, thereby increasing the range resolution properties of the hybrid waveform [6]. Frequency stepping also introduces sidelobes which are undesirable. Amplitude weights are used to reduce the area under these sidelobes while attempting to place the first null at $t=JT_c/K$. This first null, t_0, determines the resolution properties of the hybrid waveform. We define the resolution improvement factor to be

$$R = K/J \qquad (5)$$

By choosing J less than K we gain an increase in range resolution over that due to phase coding alone.

Some simplification of (4) is possible by requiring the weighting function a_k to be an even function of k with respect to the middle frequency. Manipulating the summation index k and normalizing t by T_c we have

$$|X'(t,0)| = (1/K)\ (1-|t|)\ \left|a_0 + 2\sum_{k=1}^{K-1} a_k \cos(\pi kt/J)\right| \qquad (6)$$

where $a_k=0$ if k and K are even or if k and K are odd. This last requirement allows us to consider only one function (6) for either even or odd values of K in the following discussion.

The cost function we will attempt to minimize by optimal selection of the a_k's is given by

$$c(\underline{a}) = \frac{\int_{J/K}^{1} |X'(t,0)|^2 \, dt}{\int_{0}^{1} |X'(t,0)|^2 \, dt} \qquad (7)$$

The reader will note that $c(\underline{a})$ is very similar to the usual defintion of the ISLR. Here we set the lower limit of the numerator integral equal to J/K, the desired range resolution value. As we shall see, the exact location of the first null, given a set of a_k's, will in general be slightly different from J/K due to mainlobe spreading. Also, the upper limit of both the numerator and denominator integrals is equal to 1, which corresponds to T_c, since along the delay axis, (6) is at most nonzero from -1 to 1 due to the phase coding window $1-|t|$. Finally, since (6) is an even function of t, we need only be concerned about values for $|X'(t,0)|$ in the interval $0 \leq t \leq 1$.

For a given set of a_k's, the function $|X'(t,0)|$ will have an associated PSLR, ISLR, and mainlobe spread ratio (MLSR). We define these quantities as follows,

$$PSLR = \max \frac{|X'(t,0)|^2}{|X'(0,0)|^2} \quad ; t > t_0 \qquad (8)$$

$$ISLR = \frac{\int_{t_0}^{1} |X'(t,0)|^2 \, dt}{\int_{0}^{1} |X'(t,0)|^2 \, dt} \qquad (9)$$

$$MLSR = t_0/(J/K) \qquad (10)$$

where t_0 is the <u>actual</u> location of the first null of $|X'(t,0)|$ and J/K is the <u>desired</u> location. These locations will in general be different once the a_k's have been found.

Using a conjugate gradient technique for minimizing (7), several sets of amplitude weights were generated for various values of J and K. The value of J determines the frequency step size while K determines the total frequency stepping range. For a given value of R, values for J and K are chosen to satisfy (5). Figures 3 through 6 present the evaluation of (7) through (10) assuming an optimum set of amplitude weights. Only integer values of J and K were used with the resulting points linearly connected.

The general tendencies of all four graphs are similar. For a fixed value of R, the performance improves with increasing J and thus K. This is due to the fact that with an increased number of amplitude weights we are better able to manipulate the final waveform. The MLSR also improves since with more frequencies present we have a better chance of achieving the desired null at t=J/K. We also note that as one attempts to increase the resolution improvement factor R

the performance measures worsen. This is due to the fact that as R increases while J remains fixed, the area under the sidelobes in the range $0 \leq t \leq 1$ increases. The mainlobe also tends to spread more with increasing R. Thus we see that a tradeoff exists between increased resolution and increased performance. In practical systems, acceptable levels for ISLR and PSLR values are -25dB and -30dB respectively [7]. Thus, we see that this amplitude weighting technique provides acceptable performance.

Figure 7 is a graph for comparing the performance of the optimum weighting scheme with the performance of other well known weighting functions. The uniform weighting function offers the best MLSR. The Hamming, Cos(x), and Riesz windows offer good PSLR and ISLR results but poor MLSR values. Finally, the Tukey or cosine tapered window offers good MLSR results and slightly better PSLR and ISLR values than the uniform case. The best all around performance is seen to be achieved by the optimum weighting scheme. Other popular weighting functions such as the Dolph-Chebyshev were not evaluated due to their poor MLSR performance. This is the case with all windows which exhibit a large degree of tapering at their edges.

The method used to solve for the coefficients a_k involved the minimization of the cost function using a gradient technique. An equivalent formulation of this problem can be expressed in terms of solving a matrix equation. We may express (6) in the range $0 \leq t \leq 1$ in matrix form as

$$|X'(t,0)| = (1/K) \left| [a_0\ a_1 \ldots a_{K-1}] \begin{bmatrix} 1-t \\ 2(1-t)\cos(\pi t/J) \\ 2(1-t)\cos(\pi 2t/J) \\ \vdots \\ 2(1-t)\cos(\pi(K-1)t/J) \end{bmatrix} \right|$$

$$= (1/K)\,|\,\underline{a}^T \underline{c}(t)\,| \qquad (11)$$

The expression for the cost function becomes

$$c(\underline{a}) = \frac{\underline{a}^T \underline{D}\,\underline{a}}{\underline{a}^T \underline{E}\,\underline{a}} \qquad (12)$$

where $\underline{D}$ and $\underline{E}$ are square, symmetric, matrices related to the integrals of (7). If we next define

$$\underline{A} = \underline{a}\,\underline{a}^T \qquad (13)$$

which is a square, symmetric, matrix then it can be shown that the solution to the minimization of (12) involves solving the matrix equation

$$\underline{E}\,\underline{A}\,\underline{D} = \underline{D}\,\underline{A}\,\underline{E} \qquad (14)$$

for the matrix $\underline{A}$. We note that the matrix $\underline{A}$ is one which makes the product $\underline{E}\,\underline{A}\,\underline{D}$ commute. Since all of the matrices are symmetric we note that

$$\underline{D}\,\underline{A}\,\underline{E} = [(\underline{D}\,\underline{A}\,\underline{E})^T]^T = [\underline{E}^T \underline{A}^T \underline{D}^T]^T = [\underline{E}\,\underline{A}\,\underline{D}]^T \qquad (15)$$

so that an alternate form of (14) is given by

$$\underline{E}\,\underline{A}\,\underline{D} = [\underline{E}\,\underline{A}\,\underline{D}]^T \qquad (16)$$

That is, we are looking for the matrix $\underline{A}$ which makes the product $\underline{E}\,\underline{A}\,\underline{D}$ symmetric.

Inspection of (16) initially suggests a system of K^2 nonlinear equations in K unknowns since each of the matrices $\underline{E}$, $\underline{A}$, and $\underline{D}$ are of dimension K. However, the equations involving the diagonal terms yield no information since they are identical on both sides of (16) while the equations corresponding to the elements above the main diagonal are duplicated below the main diagonal. Since there are $K(K-1)/2$ terms above or below the main diagonal, we are really attempting to solve $K(K-1)/2$ nonlinear equations in K unknowns. These equations are in fact quadratic involving all of the pairwise products of the a_k's. It should be noted, however, that some of these products will be zero. The exact ones which are zero depend on whether K is even or odd. Since the number of equations and unknowns are not equal for all K we may have no solutions, a finite number of solutions, or an infinite number of solutions for the coefficients a_k. In two dimensions, the solution describes a linear relationship between two unique coefficients indicating an infinite number of coefficient sets which minimize the cost function. As an example, the optimum weights problem for J=1 and K=4 involves the determination of two unique coefficients a_1 and a_3 which minimize the cost function $c(\underline{a})$. The coefficients a_0 and a_2 are zero since K is even. The general solution is given by the line

$$a_1 = 1.77659\ a_3 \qquad (17)$$

so that any amplitude weights satisfying this expression will yield a minimum. Figure 8 illustrates $c(\underline{a})$ for this case.

The first point to note about Figure 8 is the trough which defines the minimum value of the cost function which is -17.2 dB. In the a_1-a_3 plane the projection of this trough is described by (17). The gradient method discussed previously begins at an arbitrary point on this surface and finds its way down to the trough. As might be expected, the final point reached by this technique is highly dependent on the starting point. Since any point on the trough yields the same performance, however, the starting point is of no consequence.

IV. CONCLUSION

The purpose of this design procedure has been to improve the slant range resolution of a processor bandwidth limited SAR system. A hybrid waveform approach was used to accomplish this. Complementary phase coding was used to achieve a resolution value consistent with the bandwidth of the processor. Linear frequency stepping was added to further increase the resolution capabilities of the system. A method was shown for obtaining optimum amplitude weights for reducing the sidelobes introduced by linear frequency stepping. These amplitude weights produced results which compared favorably with practical system requirements. A numerical example was illustrated which demonstrated the nonuniqueness of these weights.

REFERENCES

1. Nathanson, F. E., Radar Design Principles, McGraw-Hill, New York, NY, 1969, Chapter 13.

2. Skolnik, M. I., Introduction to Radar Systems, Second Edition, McGraw-Hill, New York, NY, 1980, Chapter 11.

3. Sivaswamy, R., "Self-Clutter Cancellation and Ambiguity Properties of Subcomplementary Sequences," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-18, No. 2, March 1982, pp. 163-181.

4. Golay, M. J., "Complementary Series," IEEE Transactions on Information Theory, Vol. IT-7, April, 1961, pp. 82-87.

5. Tseng, C. C., and Liu, C. L., "Complementary Sets of Sequences," IEEE Transactions on Information Theory, Vol. IT-18, No. 5, September 1972, pp. 644-652.

6. Ruttenberg, K., and Chanzit, L., "High Range Resolution by Means of Pulse-to-Pulse Frequency Shifting," EASCON '68 Record, pp. 47-51.

7. Mutton, J. O., "Advanced Pulse Compression Techniques," NAECON '75 Record, pp. 141-148.

8. Harris, F. J., "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," Proceedings of the IEEE, Vol. 66, No. 1, January 1978, pp. 51-83.

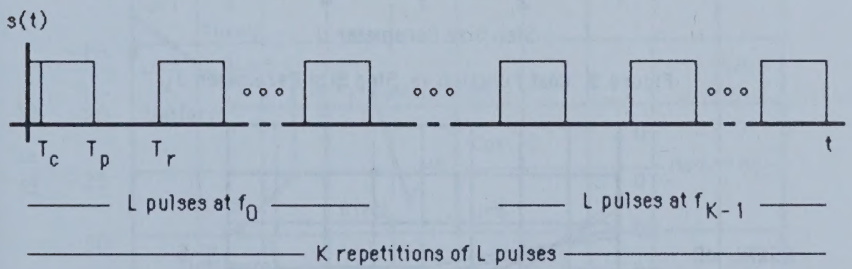


Figure 1. Frequency Stepped, Phase Coded Hybrid Waveform

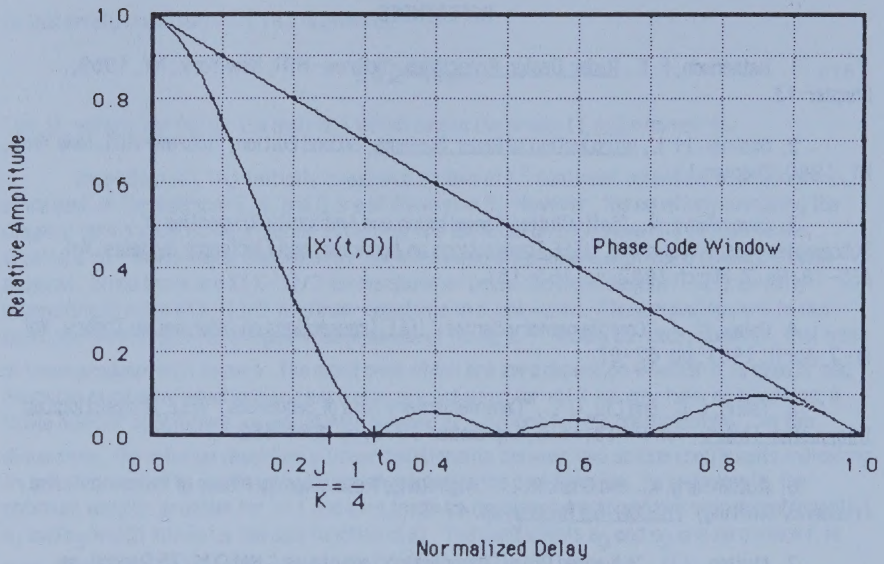


Figure 2. Zero Doppler Composite Ambiguity Diagram for J=1 and K=4

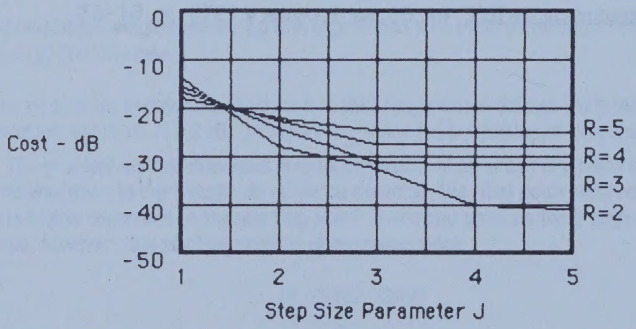


Figure 3. Cost Function vs. Step Size Parameter J

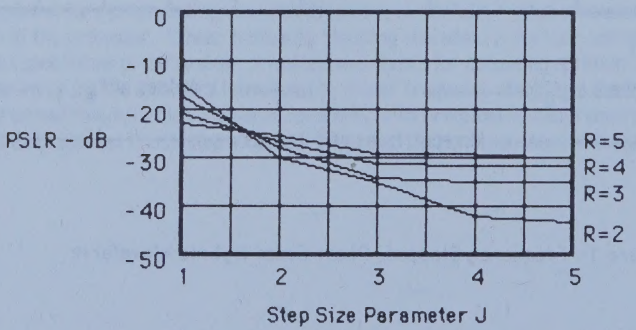


Figure 4. PSLR vs. Step Size Parameter J

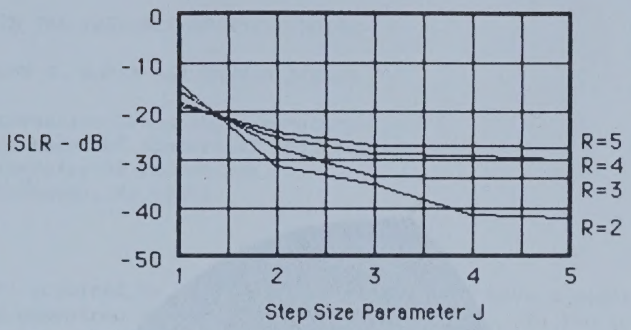


Figure 5. ISLR vs. Step Size Parameter J

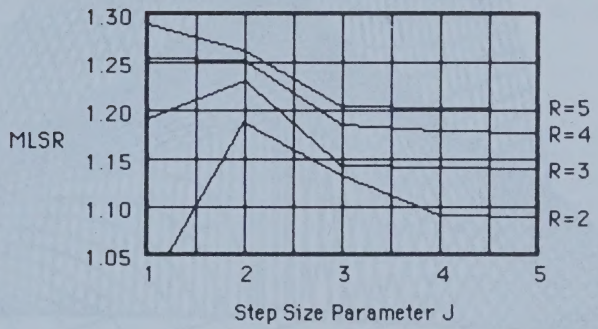


Figure 6. MLSR vs. Step Size Parameter J

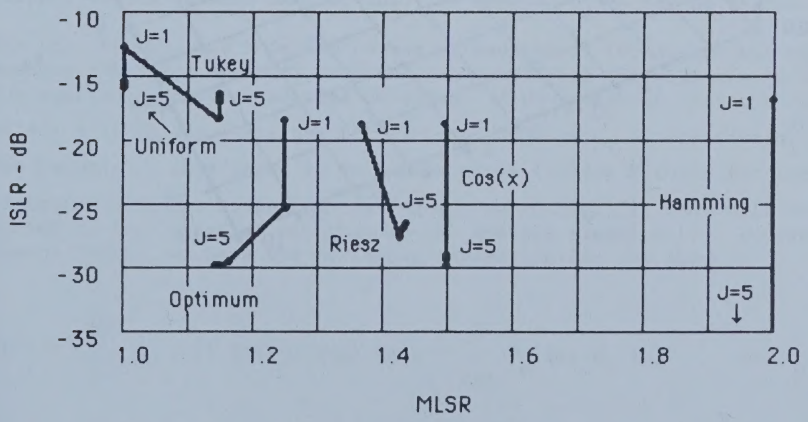


Figure 7. Amplitude Weighting Function Performance for R=4

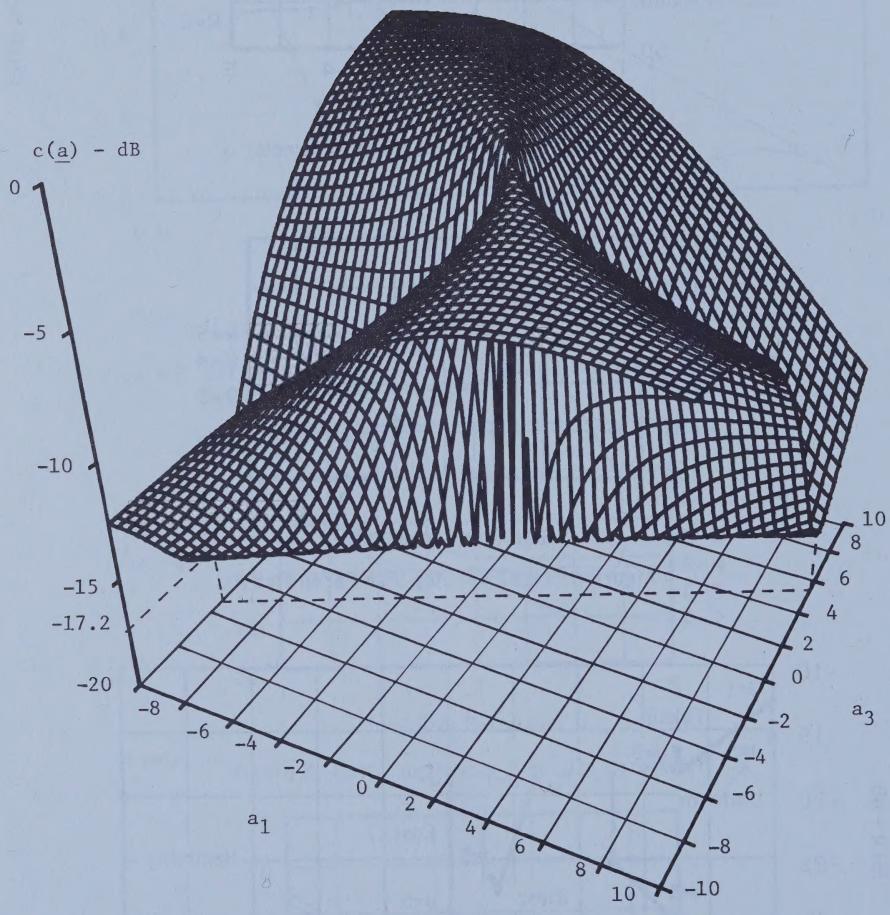


Figure 8. Cost Function for J=1 and K=4

ARRAY FILTERS FOR THE ATTENUATION OF A COHERENT UNDESIRED SIGNAL IN THE PRESENCE OF WHITE NOISE

MAGDY T. HANNA and MARWAN SIMAAN

Interactive Computing Laboratory
Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15261

ABSTRACT

Data acquired by an array of sensors often have a desired signal, a coherent undesired signal and white random noise. In the absence of the random noise, there is a family of array filters that achieve an all-pass condition for the desired signal and minimum response to the coherent undesired signal. In this paper, the member of this family that achieves the minimum output power spectrum in response to the input random noise is derived. An example illustrating the effectiveness of the filters is also presented.

I. PROBLEM FORMULATION

In many fields such as seismology, radar and sonar, arrays of sensors rather than a single sensor are used for data acquisition. Such arrays can provide the directivity needed to extract desired signals and attenuate undesired interfering signals. In most applications the undesired signals consist of coherent waveforms and random noise. Often, the measurements can be modeled by:

$$z_i(t) = x(t) + \alpha_i\, u(t - \tau_{i)} + n_i(t), \quad i = 1,\ldots,N \qquad (1)$$

where $x(t)$ and $u(t)$ are the desired and coherent undesired signals, α_i and τ_i are the relative amplitude and time delay of the undesired signal with respect to the desired one for the i th sensor, $n_i(t)$ is the random noise for that sensor, and N is the number of sensors of the array. We have assumed that $x(t)$ and $u(t)$ have the same waveforms for all sensors except for amplitude scaling and time delaying. Also we assume that the random noises $n_i(t)$'s are white and uncorrelated and having intensities σ_i^2's. The purpose of this paper is to derive array filters $F_i(\omega)$'s for the multiple input single output processing scheme of Fig. (1) that will make the output $z(t)$ as close as possible to the desired signal $x(t)$. In the frequency domain, we have the following expression for the output:

$$Z(\omega) = \left(\sum_{i=1}^{N} F_i(\omega)\right) X(\omega) + R(\omega)\, U(\omega) + \sum_{i=1}^{N} F_i(\omega)\, N_i(\omega) \qquad (2)$$

where $R(\omega)$ is the array rejection response given by:

$$R(\omega) = \sum_{i=1}^{N} \alpha_i e^{-j\omega\tau_i} F_i(\omega) \qquad (3)$$

In order to extract $X(\omega)$ we impose the constraint:

$$\sum_{i=1}^{N} F_i(\omega) = 1 \qquad (4)$$

which is an all-pass condition for the desired signal.

The most appealing minimization criterion in order to suppress the coherent undesired signal $U(\omega)$ is:

$$J(\omega) = \frac{1}{2} |R(\omega)|^2 \qquad (5)$$

The output power spectrum of the array filtering scheme of Fig. (1) in response to the input random noise is given by:

$$S_w(\omega) = \sum_{i=1}^{N} \sigma_i^2 |F_i(\omega)|^2 \qquad (6)$$

and is a reasonable criterion to minimize in order to attenuate the effect of the random noise. Therefore the array filtering problem can be stated as the multiobjective constrained minimization problem defined by the two criteria (5) and (6) and the constraint(4).

II. THE NOISE FREE PROBLEM

In the absence of the random noise, the array filtering problem reduces to the single-objective constrained minimization problem defined by (5) and (4). Such a problem has been recently shown to have a family of solutions given by the following expressions for the two mutually exclusive and comprehensive cases that can arise [1]:

Case 1:

$$\text{For } \frac{\alpha_n}{\alpha_1} e^{j\omega(\tau_n - \tau_1)} = 1 \quad , \quad n = 2, \ldots, N \qquad (7)$$

$$F = \begin{bmatrix} 1 - \sum_{n=2}^{N} c_n \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \qquad (8)$$

where $c_2, \ldots, c_N$ are arbitrary constants.

Case 2:

For $\frac{\alpha_n}{\alpha_1} e^{j\omega(\tau_n - \tau_1)} \neq 1$ for at least one n, $2 \leq n \leq N$.

$$F = \begin{bmatrix} -\sum_{n=2}^{N} c_n \left(\frac{\alpha_n}{\alpha_1}\right) e^{-j\omega(\tau_n - \tau_1)} \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \qquad (9)$$

where the constants $c_2,\ldots,c_N$ are arbitrary, subject only to the following constraint:

$$\sum_{n=2}^{N} c_n \left[1 - \left(\frac{\alpha_n}{\alpha_1}\right) e^{-j\omega(\tau_n - \tau_1)}\right] = 1 \qquad (10)$$

Moreover, it has been shown that the corresponding array rejection response is given by:

$$R(\omega) = \begin{cases} \alpha_1 e^{-j\omega\tau_1} & \text{Case 1} \\ 0 & \text{Case 2} \end{cases} \qquad (11)$$

We notice that case 2 is the one that holds for almost all frequencies, and if the α_n's are not all equal, it will hold exclusively for all frequencies. Eq. (11) implies that it has been possible to achieve an array rejection response that is zero for almost all frequencies, and if the α_n's are <u>not</u> all equal, $R(\omega)$ will be identically zero allowing perfect reconstruction of $x(t)$ in the absence of the random noise. One of the members in the above family of solutions is the Absolutely Optimum Array Filters (AOAF) which have been proved to be bounded for all frequencies and all the components have the same functional form in their dependence on frequency and the array parameters. The expression for the AOAF is given by [2,3]:

$$F_n(\omega) = \frac{\frac{1}{\alpha_n^2} - \frac{1}{N}\frac{1}{\alpha_n} e^{j\omega\tau_n} \sum_{m=1}^{N} \frac{1}{\alpha_m} e^{-j\omega\tau_m}}{\sum_{m=1}^{N} \frac{1}{\alpha_m^2} - \frac{1}{N}\left| \sum_{m=1}^{N} \frac{1}{\alpha_m} e^{-j\omega\tau_m} \right|^2}, \quad n = 1,\ldots,N \qquad (12)$$

III. ARRAY FILTERS IN THE PRESENCE OF WHITE NOISE

The freedom in the solution space of the above single-objective problem can be utilized to minimize the remaining objective (6) of the multi-objective problem as much as possible. We will consider the same previous two cases:

Case 1:

Substituting the solution (8) into criterion (6) we get[1]:

$$S_w(\omega) = \sigma_1^2 |1 - \mu' c|^2 + c^{*'} \sigma c \tag{13}$$

where

$$c = (c_2\ c_3 \ldots c_N)' \tag{14}$$

and

$$\sigma = \text{diag}\ \{\sigma_2^2, \ldots, \sigma_N^2\} \tag{15}$$

Determining the vector c that minimizes (13) and substituting into (8) we get the following solution:

$$F_n(\omega) = \frac{\frac{1}{\sigma_n^2}}{\sum_{m=1}^{N} \frac{1}{\sigma_m^2}} \quad , n = 1, \ldots, N \tag{16}$$

Case 2:

Substituting the solution (9) into criterion (6) we get:

$$S_w(\omega) = \sigma_1^2\, c^{*'} dd^{*'} c + c^{*'} \sigma c \tag{17}$$

where

$$d = \left(\frac{a_2}{a_1} \quad \frac{a_3}{a_1} \ldots \frac{a_N}{a_1}\right)' \tag{18}$$

and

$$a_n = \alpha_n e^{j\omega\tau_n} \quad , n = 1, \ldots, N \tag{19}$$

---

[1] The superscript * means complex conjugate and the ' means transpose.

The problem reduces to determining the vector c that minimizes (17) subject to constraint (10) which can be written compactly using (14), (18) and (19) as

$$(\mu - d)^{*'} c = 1 \tag{20}$$

where

$$\mu = (1\ 1\ \ldots 1)' \tag{21}$$

The solution of this problem is given by [4]:

$$F_n(\omega) = \frac{\frac{1}{\sigma_n^2} - \left[\sum_{m=1}^{N} \left(\frac{\alpha_m}{\sigma_m}\right)^2\right]^{-1} \frac{\alpha_n}{\sigma_n^2} e^{j\omega\tau_n} \sum_{m=1}^{N} \frac{\alpha_m}{\sigma_m^2} e^{-j\omega\tau_m}}{\sum_{m=1}^{N} \left| \frac{1}{\sigma_m^2} - \left[\sum_{m=1}^{N} \left(\frac{\alpha_m}{\sigma_m}\right)^2\right]^{-1} \frac{\alpha_m}{\sigma_m^2} e^{-j\omega\tau_m} \right|^2}, n = 1,\ldots,N \tag{22}$$

The solution obtained in this section is optimal with respect to the first criterion, but suboptimal with respect to the other since it has been confined to lie in the solution space of the first.

IV. AN ILLUSTRATIVE EXAMPLE

The 8 input signals $z_n(t)$' s shown in Fig (4) have been generated by adding the signals and noise as discribed below:

(1) The desired signal x(t) shown in Fig. (2)
(2) The undesired signal u(t) shown in Fig. (3), after being amplitude scaled and time shifted by the following α_n's and τ_n's:

n	1	2	3	4	5	6	7	8
α_n	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
τ_n in m sec.	0	14	26	36	44	50	54	56

(3) Eight zero-mean random noise sequences having the same variance and generated by a random number generator

The undesired signal u(t) has the same energy as x(t) and each random noise sequence is 6 dB higher in energy than x(t). The same set of input signals has been processed by the following methods:

(1) The new filters derived in this paper
(2) The Absolutely Optimum Array Filters (AOAF) that are optimal for

the noise free case.

(3) A weighted delay-and-sum technique defined by:

$$F_n(\omega) = \frac{\frac{1}{\alpha_n^2}}{\sum_{m=1}^{N} \frac{1}{\alpha_m^2}} \quad ,n = 1,\ldots,N \qquad (23)$$

Fig (5) shows the outputs for the above 3 methods. It is clear that the new filters are superior to the others.

V. CONCLUSION

Array filters have been derived for extracting a desired signal from the measurements of an array of sensors having a desired signal, a coherent undesired signal and white random noise. The freedom in the solution space of the single-objective constrained minimization problem-defined by the squared magnitude of the array rejection response as an objective and the all-pass condition for the desired signal as a constraint-has been utilized in minimizing the output power spectrum due to the random noise input. The superior behavior of the derived filters has been shown by an illustrative comparative example.

REFERENCES

1. M.T. Hanna and M. Simaan, "A Closed Form Solution to a Quadratic Programming Problem in Complex Variables," accepted for publication in the Journal of Optimization Theory and Applications.

2. M.T. Hanna and M. Simaan, "Absolutely Optimum Filtering for Sensor Array Data, "Proceedings of the 27th Midwest Symposium on Circuits and Systems, Morgantown, WV, June 11-12, 1984.

3. M.T. Hanna and M. Simaan, "On the Theory of Generalized and Absolutely Optimum Filters for Sensor Arrays," submitted for publication.

4. M.T. Hanna and M. Simaan, "Minimum Reject Response Array Filters in the Presence of White Noise," submitted for publication.

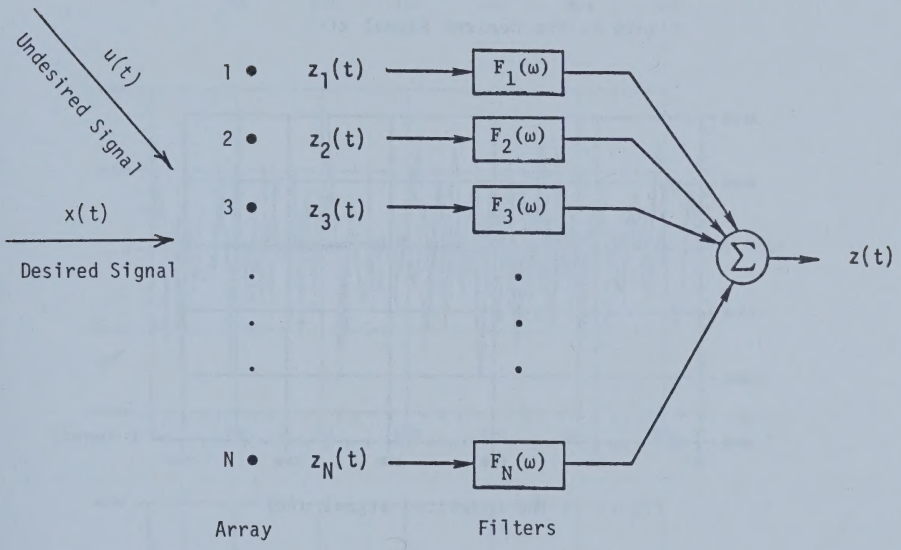


Figure 1: Array Sensors and Filters Configuration

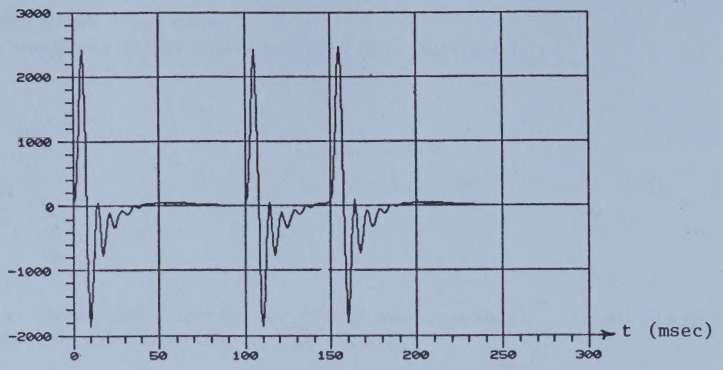


Figure 2: The Desired Signal x(t)

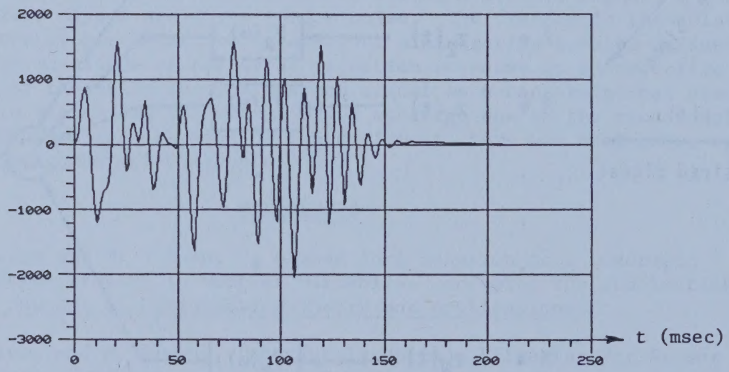


Figure 3: The undesired signal u(t)

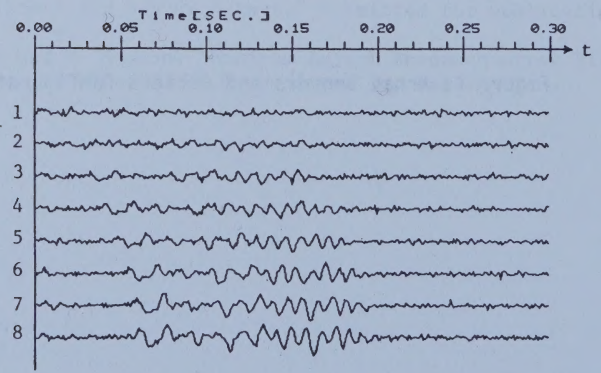


Figure 4: The 8 input signals $z_i(t)$, i=1, ..,8

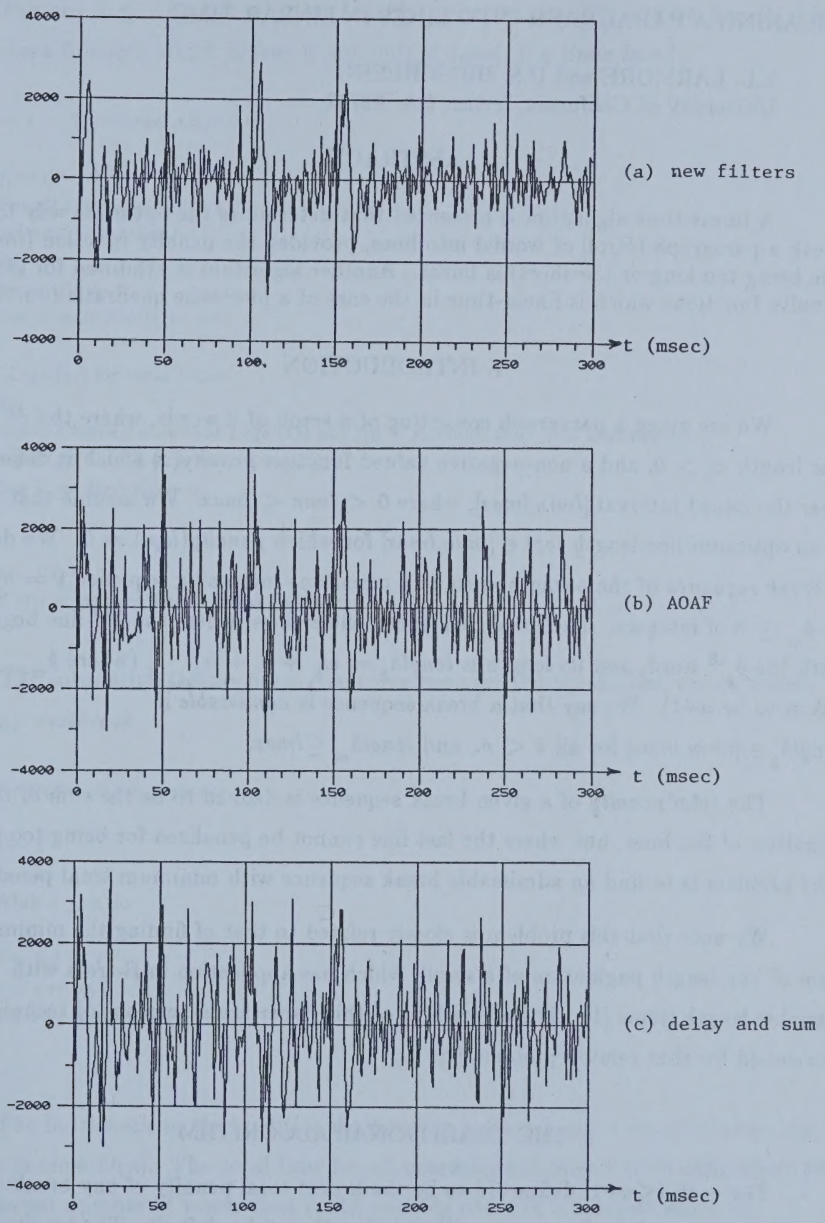


Figure 5: The Output of the Array Filters.
(a) new filters
(b) AOAF
(c) delay and sum

BREAKING A PARAGRAPH INTO LINES IN LINEAR TIME

L.L. LARMORE and D.S. HIRSCHBERG
University of California, Irvine, CA 92717

ABSTRACT

A linear-time algorithm is presented that determines the optimum way to break a paragraph (scroll of words) into lines, provided the penalty function (for a line being too long or too short) is linear. Another algorithm is exhibited for general penalty functions which is linear-time in the case of a piecewise quadratic function.

1. INTRODUCTION

We are given a paragraph consisting of a scroll of n words, where the i^{th} word has length $w_i > 0$, and a non-negative valued function $penalty(x)$ which is defined over the closed interval $[lmin,lmax]$, where $0 < lmin < lmax$. We assume that there is an optimum line length $lopt \in [lmin,lmax]$ for which $penalty(lopt) = 0$. We define a *break sequence* of the paragraph to be a monotone increasing sequence $1 = b_1, b_2, \ldots b_m \leq n$ of integers. The break sequence defines lines, where the k^{th} line begins with the b_k^{th} word, and its length is $length_k = w_{b_k} + \ldots + w_{b_{k+1}-1}$ (where b_{m+1} is taken to be $n+1$). We say that a break sequence is *admissable* if $length_k \in [lmin,lmax]$ for all $k < m$, and $length_m \leq lmax$.

The *total penalty* of a given break sequence is defined to be the sum of the penalties of the lines, but where the last line cannot be penalized for being too short. The problem is to find an admissable break sequence with minimum total penalty.

We note that this problem is closely related to that of finding the minimum sum of key length pagination of a scroll, which has application in B-trees with variable-length items [1]. The algorithms of this paper are extensions of techniques developed for that related problem [2].

2. THE TRADITIONAL ALGORITHM

For each $i \leq n+1$, define $f[i]$ to be the lowest total penalty of any break sequence of the subscroll $w_i \ldots w_n$. We let $f[n+1] = 0$ by default. The traditional algorithm (see, for example, [3]) uses dynamic programming.

This research was supported in part by National Science Foundation Grant MCS-82-00362. Authors' address: Department of Information and Computer Science, University of California, Irvine, CA 92717.

For any $1 \leq i \leq j \leq n+1$, let $Line(i,j) = w_i + \ldots + w_{j-1}$, and let $Legal(i,j)$ be the boolean function which is true if and only if $Line(i,j) \in [lmin,lmax]$.

Algorithm 1 -- Traditional Algorithm

```
Step 1. f[n+1] ← 0
Step 2. For all i from n downto 1 do
  If Line(i,n+1)≤lopt then
    begin
      Step 3. f[i] ← 0
      Step 4. nextbreak[i] ← n+1
    end
  else if Legal(i,j) for some j then
    begin
      Step 5. Choose j such that Legal(i,j) and f[j] + penalty(Line(i,j)) is minimized
      Step 6. f[i] ← f[j] + penalty(Line(i,j))
      Step 7. nextbreak[i] ← j
    end
  else
    Step 8. f[i] ← ∞
Step 9. If f[1] < ∞ then Define_break_sequence
```

The subroutine Define_break_sequence recovers the breakpoint vector b from the array *nextbreak*.

Subroutine Define_break_sequence

```
Step 1. s ← 1
Step 2. t ← 1
Step 3. While t ≤ n do
  begin
    Step 4. b[s] ← t
    Step 5. s ← s+1
    Step 6. t ← nextbreak[t]
  end
```

The bottleneck in the traditional algorithm is Step 5, since all other steps can be done in time $O(n)$. The total time for all executions of Step 5 is $O(nM)$, where M is the largest number of words that could possibly occur in a line (we could set $M = lmax/W$, where W is the minimum value of w_i). If M is bounded, then the traditional algorithm is linear. However, if we consider a class of problems in which M, as well as n, grows then the traditional algorithm is no longer linear.

3. LINEAR PENALTY FUNCTIONS

We consider first the case that *penalty* is linear, i.e., that there exists a constant C such that, for all $x \in [lmin,lmax]$, $penalty(x) = C(x\text{–}lopt)$. The constant may be positive, negative, or zero. To avoid possible deadlock in Step 11, we let $penalty(x) = \infty$ if $x \notin [lmin,lmax]$.

We will use dynamic arrays *leftlow* and *rightlow*, which will have pointer (i.e., integer) values, and dynamic arrays f and g, which will have penalty values. The array f is as in the traditional algorithm, and g will be a modified array which will always satisfy the equation $g[k] = f[k] + CLine(1,k)$. At any given time, $rightlow[k]$ will be the smallest $l > k$ such that $g[l] \leq g[k]$, and $leftlow[k]$ will be the largest $l < k$ such that $g[l] < g[k]$. A running pointer r will also be used.

Algorithm 2 -- Linear Penalty Algorithm

```
Step 1. g[n+2] ← -∞
Step 2. f[n+1] ← 0
Step 3. g[n+1] ← CLine(1,n+1)
Step 4. r ← n+1
Step 5. rightlow[n+1] ← n+2
Step 6. For all i from n downto 1 do
  begin
    If Line(i,n+1)≤lopt then
      begin
        Step 7. f[i] ← 0
        Step 8. nextbreak[i] ← n+1
      end
    else
      begin
        Step 9. While Line(i,r) > lmax do    r ← r-1
        Step 10. While leftlow[r] defined and Legal(i,leftlow[r]) do    r ← leftlow[r]
        Step 11. f[i] ← f[r] + penalty(Line(i,r))
        Step 12. If f[i] < ∞ then nextbreak[i] ← r
      end
    Step 13. g[i] ← f[i] + CLine(1,i)
    Step 14. k ← i+1
    Step 15. While g[k] > g[i] do
      begin
        Step 16. leftlow[k] ← i
        Step 17. k ← rightlow[k]
      end
    Step 18. rightlow[i] ← k
```

end (of Step 6)
Step 19. If $f[1] < \infty$ then Define_break_sequence

Proof of Correctness

Assume the correctness of the traditional algorithm. We show that the Linear Penalty Algorithm simulates the traditional algorithm. We need three lemmas.

Lemma A. Fix i. Among the set of all j such that $Legal(i,j)$, that j which minimizes $g[j]$ also minimizes $f[j] + penalty(Line(i,j))$.

Proof. $f[j] + penalty(Line(i,j)) - g[j] = penalty(Line(i,j)) - CLine(1,j) = -C(lopt + Line(1,i))$, which is constant for fixed i. □

Lemma B. The following loop invariant holds after each execution of Step 18 of the linear penalty algorithm:

LB1: For all $i \leq s \leq n+1$, $rightlow[s] = t$, where $t > s$ is the smallest value such that $g[t] \leq g[s]$.

LB2: For all $i < s \leq n+1$, $leftlow[s] = t$, where $i \leq t < s$ is the largest value such that $g[t] < g[s]$, provided such a t exists. Thus, in this case, for all $leftlow[s] < j < s$, $g[j] \geq g[s]$. Otherwise, $leftlow[s]$ is undefined.

Proof. By induction on i. Initially, i.e., before Step 6 iterates at all, we can take $i = n+1$. Then, LB2 holds vacuously, while LB1 holds by Steps 1 and 5.

Our inductive hypothesis is that LB holds for all values larger then i. Thus, before execution of Step 15, for all $i+1 \leq s \leq n+1$:

(a) $rightlow[s] = t$, where $t > s$ is the smallest value such that $g[t] \leq g[s]$.

(b) $leftlow[s] = t$, where $i+1 \leq t < s$ is the largest value such that $g[t] < g[s]$, provided such a t exists. Otherwise, $leftlow[s]$ is undefined.

Thus, we need to prove only that, after execution of Step 18, $rightlow[i]$ has the correct value, and that $leftlow[s] = i$ if $i < s \leq n+1$ and $g[i] < g[s]$ and $leftlow[s]$ was undefined before Step 15.

Let $k_0, \ldots k_m$ be the sequence of values of k produced in Steps 14 and 17, i.e., $k_0 = i+1$, and $k_{l+1} = rightlow[k_l]$ for $0 \leq l < m$. Note that $g[k_m] \leq g[i]$, and

$g[k_l] > g[i]$ for all $0 \leq l < m$.

Sublemma. For all $0 \leq l \leq m$, and for all $i < s \leq k_l$, $g[s] \geq g[k_l]$. Furthermore, if $l \geq 1$ and $s < k_l$, $g[s] \geq g[k_{l-1}]$.

Proof. By induction on l. For $l = 0$, we have $s = k_0 = i+1$. Assume the sublemma holds for $l-1$. If $i < s \leq k_l$ then, because $k_l = rightlow[k_{l-1}] > k_{l-1}$ by LB1, either (I) $s \leq k_{l-1}$, or (II) $k_{l-1} < s < k_l$, or (III) $s = k_l$.

If (I) then, by the inductive hypothesis, $g[s] \geq g[k_{l-1}]$.

If (II) then we have the following. $k_l = rightlow[k_{l-1}]$ by Step 17. By LB1, $g[k_l] \leq g[k_{l-1}]$ and for any s between k_{l-1} and k_l, $g[s] > g[k_{l-1}]$.

In either case we are done, since $g[k_l] \leq g[k_{l-1}]$. Case (III) is trivial. □

Proof of Lemma B, continued. By the sublemma, all $i < s \leq k_l$ are unsuitable values for $leftlow[k_l]$ since $g[s] \geq g[k_l]$. Therefore, for any $l < m$, $leftlow[k_l]$ should be assigned the value i if $g[i] < g[k_l]$. Step 16 does exactly that. We now show that $rightlow[i]$ should be assigned the value k_m, that is, $g[k_m] \leq g[i]$ and $g[s] > g[i]$, for $i < s < k_m$.

Since Step 15 no longer iterates when $k = k_m$, $g[k_m] \leq g[i]$. We now show that $g[s] > g[i]$ for all $i < s < k_m$. This is true if $m = 0$ since $k_0 = i+1$; otherwise $g[s] \geq g[k_{m-1}]$ by the sublemma, and $g[k_{m-1}] > g[i]$ because Step 15 still iterates when $k = k_{m-1}$. Thus, Step 17 makes the correct assignment to $rightlow[i]$. □

Lemma C. After execution of Step 10, one of the following conditions holds:

(I) There is no $j \leq n+1$ such that $Legal(i,j)$, and r is the largest possible value of $j \leq n+1$ such that $Line(i,j) \leq lmax$.

(II) Among all j such that $Legal(i,j)$, r is the choice of j which minimizes $g[j]$.

Proof. By induction on i. Condition (I) holds before the first iteration of Step 6 (consider $i = n+1$). We define a loop invariant:

LC1: For all $r < j \leq n+1$ such that $Line(i,j) \leq lmax$, $g[j] \geq g[r]$.

LC2: If $r < n+1$, $Line(i,r+1) \geq lmin$.

We show the invariant holds initially i.e., before Step 6, and that it is preserved by each execution of Steps 9 or 10, as well as when i is decremented in Step 6.

Initially, LC holds vacuously. Decrementing i cannot cause LC to fail, because $Line(i,j)$ is monotone decreasing on the first argument. An iteration of Step 9 preserves LC2 because, immediately after any such iteration, $Line(i,r+1) > lmax \geq lmin$. Also, if Step 9 iterated one or more times, LC1 holds vacuously.

We now show that an iteration of Step 10 preserves LC. r' denotes the value of r after the iteration.

If LC holds before an iteration then, by LC1, $g[j] \geq g[r]$ for all $r < j \leq n+1$ such that $Line(i,j) \leq lmax$. In order for Step 10 to iterate, $leftlow[r]$ is defined and $Line(i,leftlow[r]) \geq lmin$.

LC2 will hold after the iteration, since $Line(i,leftlow[r]+1) > Line(i,leftlow[r]) \geq lmin$ and thus $Line(i,r'+1) \geq lmin$.

To prove that LC1 is preserved, consider j such that $r' = leftlow[r] < j \leq n+1$ and $Line(i,j) \leq lmax$. We need to show that $g[j] \geq g[leftlow[r]] = g[r']$. If $r < j \leq n+1$ then $g[j] \geq g[r]$ by LC1 and $g[r] > g[r']$ by LB2. If $r' = leftlow[r] < j \leq r$ then $g[j] \geq g[r] > g[leftlow[r]]$ by LB2.

Thus, LC is loop invariant.

Consider the case when $Legal(i,r)$ after Step 9 has executed. By LC1, $g[j] \geq g[r]$ for any $j > r$ such that $Legal(i,j)$. If $j < r$ and $Legal(i,j)$ then, since Step 10 has ceased iterating, either $leftlow[r]$ is undefined, or $Line(i,leftlow[r]) < lmin$ which implies by monotonicity of $Line$ that $leftlow[r] < j < r$. In either case, by LB2, $g[j] \geq g[r]$. Thus case (II) of Lemma C holds.

Otherwise, not $Legal(i,r)$ after Step 9 has executed. In this case, $Line(i,r) \leq lmax$ since Step 9 did not iterate again, and therefore $Line(i,r) < lmin$ since otherwise $Legal(i,r)$. Thus, Step 10 cannot iterate by the monotonicity of $Line$ and the fact that if $leftlow[r]$ is defined then $leftlow[r] < r$. Now, either Step 9 iterated once or more, or it did not iterate. If Step 9 iterated, Case (I) of Lemma C holds, since then $Line(i,r+1) > lmax$ (and Step 10 did not iterate). If Step 9 did not

iterate, r has the same value as it did after execution of Step 10 of the previous iteration of the main loop. By the inductive hypothesis, Lemma C held for $i+1$. Case (II) cannot have held, because then $Line(i,r) > Line(i+1,r) \geq lmin$, which would imply that $Legal(i,r)$. Thus Case (I) of Lemma C held for $i+1$. Since $Line$ is monotone decreasing on its first parameter, Case (I) of Lemma C then holds for i. □

We now prove the correctness of the Linear Penalty Algorithm, assuming that the traditional algorithm is correct. We need to show that Steps 9 through 12 of the Linear Penalty Algorithm are equivalent to Steps 5 through 8 of the traditional algorithm. By Lemmas A and C (only case C(II) applies since we are within the *else if* clause), the value of r after execution of Step 10 is the same as the value of j chosen in Step 5 of the traditional algorithm, provided such a legal j exists. Step 11 is equivalent to both Steps 6 and 8 of the traditional algorithm, while Step 12 is equivalent to Step 7 of the traditional algorithm. □

Proof of Linear Time

We can compute $Line(1,i)$ for all i in $O(n)$ time. Then $Line(i,j)$ (and hence $Legal(i,j)$) can be computed in $O(1)$ time. Steps 1 through 5 take time $O(1)$. Steps 7, 8, 11, 12, 13, 14 and 18 execute no more than once for each of the $O(n)$ iterations of Step 6. Step 19 takes $O(n)$ time.

Each iteration of Steps 9 or 10 decreases r. Thus, the total number of such iterations cannot exceed n.

Note, by LB2, the value of $leftlow[k]$, once defined, is never redefined. Thus, Step 16 can execute at most n times. It follows that the total number of iterations of Step 15, over all iterations of Step 6, is at most n. □

4. GENERAL CONCAVE PENALTY FUNCTIONS

We say that a function $p(x)$ is *concave* if, for any $x < y < z$ in its domain, $(z-x)p(y) \leq (y-x)p(z) + (z-y)p(x)$. For example, any quadratic function with non-negative leading coefficient is concave.

We now consider the breaksequence problem where $penalty(x)$ is non-negative and concave for $x \in [lmin,lmax]$. As before, $penalty(x) = \infty$ for $x \notin [lmin,lmax]$, and

there is no penalty for the last line if its length does not exceed *lopt*.

The time bottleneck in the General Concave Algorithm given below is the evaluation of the Boolean function *Bridge*. All other parts of the algorithm run in linear time, and *Bridge* needs to be evaluated $O(n)$ times. If *penalty* is quadratic, *Bridge* can be evaluated in $O(1)$ time, and hence the entire algorithm is linear. Generally, *Bridge* can be evaluated in $O(\log M)$ time by binary search, making the entire algorithm $O(n \log M)$. We leave open the possibility that a faster general algorithm exists.

Notation. We let $F(i,j) = f[j]+penalty(Line(i,j))$, the least cost of a paragraph beginning at the i^{th} word whose *second* line begins at the j^{th} word.

The Boolean function Bridge. $Bridge(j,k,l)$ is defined for $1 < j < k < l \leq n+1$. If *true*, it means that k need not be considered as a choice for $nextbreak[i]$ for any "future" i (i.e., $i < j$), since either j or k is always (i.e., for any i) at least as good a choice. Formally, for the algorithm to run correctly, it suffices that *Bridge* satisfy the following two conditions:

Br1: If $1 \leq i < j < k < l \leq n+1$ such that $Legal(i,k)$ and $Bridge(j,k,l)$, then $F(i,j) \leq F(i,k)$ or $F(i,l) \leq F(i,k)$.

Br2: If $1 \leq i < j < k < l \leq n+1$ such that $Legal(i,k)$ and not $Bridge(j,k,l)$, then $F(i,k) \leq F(i,j)$ or $F(i,k) \leq F(i,l)$.

There is an allowed ambiguity in the definition of *Bridge*. Any function that satisfies Br1 and Br2 will work. For example, we can define $Bridge(j,k,l) = true$ iff for some i, $F(i,k) < F(i,j)$ and $F(i,k) < F(i,l)$. At most M possible values of i need be examined. Since *penalty* is concave, *Bridge* can be computed in $O(\log M)$ time by using binary search.

Quadratic Case. Suppose that $penalty(x) = ax^2 + bx + c$ for $x \in [lmin,lmax]$, where $a \geq 0$. Then for any $j < k < l$, let $Bridge(j,k,l)$ be *true* if and only if the following two conditions hold:

Q1: $f[k] + penalty(lmax\text{–}Line(k,l)) \geq f[j] + penalty(lmax\text{–}Line(j,l))$

Q2: $Line(j,l)f[k] \geq Line(j,k)f[l] + Line(k,l)f[j] + a\,Line(j,k)\,Line(j,l)\,Line(k,l)$

We note that Q1 and Q2 can both be computed in $O(1)$ time. Thus, we have a linear time algorithm for the case of a quadratic penalty function.

Data structure. We will make use of an input-restricted deque S of integers. Integers can be deleted from both the top and bottom ends of S, but can only be inserted to the top end. The following operators on S will be needed:

Functions:	$\|S\|$	= current cardinality of S
	Top	= value of the top element of S
	Bottom	= value of the bottom element of S
	2Top	= value of the second from the top element of S
	2Bottom	= value of the second from the bottom element of S
Procedures:	*Pop*	delete the top element of S
	Drop	delete the bottom element of S
	Push(x)	insert x at the top of S

Algorithm 3 -- Concave Penalty Function

```
Step 1. f[n+1] ← 0
Step 2. S ← Λ (empty list)
Step 3. edge ← n+1
Step 4. For i from n downto 1 do
  begin
    Step 5. While S nonempty and Line(i,Bottom) > lmax do    Drop
    Step 6. While Line(i,edge) ≥ lmin do
      begin
        Step 7. While S nonempty and F(i,edge) ≤ F(i,Top) do    Pop
        Step 8. While |S| ≥ 2 and Bridge(edge,Top,2Top) do    Pop
        Step 9. If Line(i,edge) ≤ lmax then Push(edge)
        Step 10. edge ← edge-1
      end
    Step 11. While |S| ≥ 2 and F(i,2Bottom) ≤ F(i,Bottom) do    Drop
    If Line(i,n+1) ≤ lopt then
      begin
        Step 12. nextbreak[i] ← n+1
        Step 13. f[i] ← 0
      end
    else if S nonempty then
      begin
        Step 14. nextbreak[i] ← Bottom
        Step 15. f[i] ← F(i,Bottom)
      end
    else (i.e., S = Λ)
```

Step 16. $f[i] \leftarrow \infty$
end (of Step 4)
Step 17. If $f[1] < \infty$ then Define_break_sequence

5. PIECEWISE CONCAVE PENALTY FUNCTION

If the penalty function is piecewise concave, the algorithm can be generalized, using one deque for each concave piece. In the case where the function is piecewise linear or piecewise quadratic, the running time is $O(n\Gamma)$.

The method is essentially to use independent copies of the general concave algorithm, one for each concave piece of the penalty function. These procedures meet once during each iteration of the main For loop to exchange information and decide which one has the best value for *nextbreak*[*i*].

Assume there are Γ intervals, $I_\gamma = [lmin_\gamma, lmax_\gamma]$ for each $1 \leq \gamma \leq \Gamma$, which are disjoint or meet only at endpoints. We shall assume that $penalty_\gamma(x)$ is defined and concave for all $x \in I_\gamma$, and is ∞ for all x outside of that interval, and that $penalty(x)$ agrees with $penalty_\gamma(x)$ on each I_γ, and is ∞ outside the union of the I_γ.

For each γ, we maintain an input-restricted deque S_γ, along with all the operators we used in the general concave algorithm.

REFERENCES

[1] Diehr, G. and Faaland, B. Optimal pagination of B-trees with variable-length items. *Comm. ACM 27*, 3 (March 1984), 241-247.

[2] Larmore, L.L. and Hirschberg, D.S. Efficient optimal pagination of scrolls. Technical Report, Univ. of Calif. Irvine ICS Dept., April 1984.

[3] Knuth, D.E. and Plass, M.F. Breaking paragraphs into lines. *Software - Practice & Experience* (1981), 1119-1184.

A FAST LINE INTERSECTION ROUTING METHOD FOR OPTIMAL WIRINGS

JAMES P. COHOON
University of Virginia
Charlottesville, Virginia 22903

Abstract: We present theorems that provide limits on the search area for an optimal wirings with respect to bend minimization and length minimizations. We also describe a new rectilinear routing algorithm that exploits these theorems to quickly produce optimal solutions.

INTRODUCTION

Design Automation has steadily introduced software tools to aid the circuit design process. The contribution presented here is a new line intersection approach for the *Routing Problem*. Our approach operates on a gridless, arbitrarily shaped, continuous, multilayer medium with interconnections that have arbitrarily specified width and separation. The approach has the following characteristics: guarantees an optimal solution if a wiring exists; good run-time and space complexities that are independent of wire length; flexible design rule specifications; and the ability to readily incorporate heuristics for better run-time performance. As time complexity is independent of wiring length, for some problem instances the approach is arbitrarily faster than previous routers. These characteristics are achieved by exploiting two graph-theoretic theorems that are proven below. The theorems provide limits on the search area for an optimal solution.

Problem Formulation

A *circuit medium* is a continuous region, possibly multilayered, whose boundaries are arbitrarily shaped. A *via* is a feed-through from one layer of a circuit medium to adjoining layers.

A digital device is often decomposed into a collection of functional units or *modules*. The modules occupy portions of the circuit medium. Individual modules may have different sizes and shapes. The only restriction for a module is that all its boundary edges must be rectilinear. A similar restriction is applied to the layout of wires. Though they decrease generality, such restrictions allow for the compact representation of modules and wires, and typically reflect the capabilities of automatic fabrication equipment.

Various subsets of the modules may require that information be shared for the operation of the digital device. Each such subset is a *net* and the information is a *signal*. A *terminal* lies on the boundary of a module and is an interconnection point for a net. A signal is shared by interconnecting the terminals that compose a net with wires. Often nets are decomposed into a tuple of order terminal pairs. The terminal pairs are then wired one at a time in tuple order.

A *routing region* is that portion of the circuit medium unoccupied by modules.

A *routing segment* implements all or part of an interconnection. It has direction, position, and width on a specified layer of the routing region. The position of a routing segment is denoted by its endpoints. Formally, a *wire* is a series of routing segments that connects two terminals. Two successive routing segments of a wire share either a common endpoint on the same layer or have endpoints that share a via. A *design rule* is a restriction on the way wires are positioned, formed, or interact. Restrictions are imposed to guarantee the presence or absence of some electrical condition.

The Routing Problem has the following formal specification:

> **Input:** A circuit medium, a set of modules, a routing region, a net composed of two terminals, a set of vias, a set of design rules, and an optimization criterion of either bend minimization or length minimization.
> **Output:** A set of locally optimal routing segments that implements all nets.

Present Approaches

There are two primary approaches to the Routing Problem: maze routers [6,7] and line search routers [3,4]. For additional background and bibliography see [5,9]. Maze routers find optimal solutions under a wide variety of optimization criteria. However, they have extremely large memory requirements to represent reasonable-sized circuits and can be slow to find long

interconnections. Line search routers do not store the entire circuit medium and thus have more reasonable storage requirements than maze routers; they also have good run-time performance for up to medium size problems; and their solutions are typically close to the minimal number of bends. However, line search routers do not guarantee finding a solution and their performance degrades and their storage requirements increase with dense circuits [9].

Line Intersection Approach

Our line intersection approach is a synthesis of the maze and line search approaches. This synthesis allows our router to possess the advantages of both methods without their disadvantages.

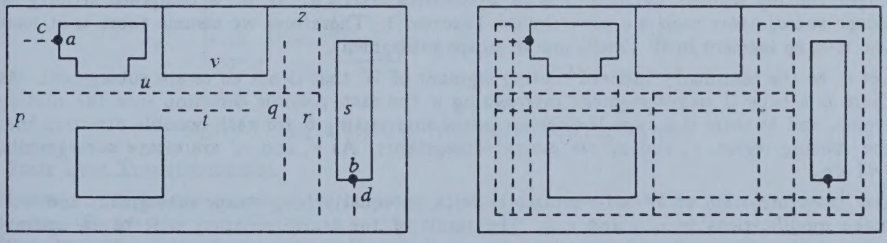
The line intersection approach examines a limited subregion, S, of the routing region. S is organized as a collection of possible routing segments and is guaranteed to contain an optimal wiring if such a wiring exists. The routing segments in S are examined in either a *breadth-first* or *shortest-path search* order (depending upon the optimization criterion). These efficient search techniques enable the router to quickly determine an occurrence of an optimal wiring.

In subsequent sections, we present some additional terminology; specify the input form; demonstrate the existence of subregion S for either a single layer or multilayer circuit medium; and describe a single layer version of a router that connects a single pair of terminal endpoints. The design rule that our router employs concerns the width and separation of routing segments – all wires are 1 width unit wide and are at least 1 separation unit apart. In [1], we discuss generalizations and variations that allow a multilayer router, a set of terminal pairs, and arbitrary wire separations and widths.

INPUT SPECIFICATION AND NECESSARY TERMINOLOGY

The modules and circuit medium are represented as a set of line segments, M, that form their boundaries. Associated with each module and circuit perimeter boundary segment is an indicator on which side the interior of the module or the circuit medium respectively lies. Without loss of generality, we make two assumptions. Firstly, we assume that no two modules are physically adjacent. If two modules are physically adjacent then their two representations can be collapsed into one representation with the removal of the common boundary segments. The result is that a module is bordered on each of its sides by either the routing region or the perimeter of the circuit medium. Secondly, we assume that every boundary segment is connected at each endpoint to a perpendicular boundary segment that is also given as input. For objects a single unit in length or width, the endpoints satisfy this requirement.

A horizontal segment $u=((x_u,y_u),(w_u,y_u))$ *covers* horizontal boundary segment $v=((x_v,y_v),(w_v,y_v))$ iff the intersection of the closed intervals $[x_u,w_u]$ and $[x_v,w_v]$ is nonempty. A definition follows analogously for a vertical covering. In Figure 1.a, t and u both cover (horizontally) v, and q and r both cover (vertically) s.



(a) – Covers, Limiting and Escape Segments (b) – All Escape Segments

Figure 1 – Terminology Examples

A horizontal segment $u=((x_u,y_u),(w_u,y_u))$ in the routing region is *maximal* iff neither of the points (x_u-1,y_u) and (w_1+1,y_u) are in the routing region. The offset of 1 represents the basic unit of separation. A definition follows analogously for a maximal vertical segment. In Figure 1.a, u and r are maximal and t and q are nonmaximal.

The vertical boundary segment that contains the point (x_u-1,y_u) for maximal segment $u=((x_u,y_u),(w_u,y_u))$ is the *left limiting* boundary segment of u. The vertical boundary segment that contains the point (w_u+1,y_u) is the *right limiting* boundary segment of u. Definitions follow analogously for *lower limiting* and *upper limiting* horizontal boundary segments. In Figure 1.a, p is a left limiting segment for u and z is an upper limiting segment for r.

A horizontal line segment $u=((x_u,y_u),(w_u,y_u))$ in the routing region is a *type I escape segment* for horizontal boundary segment $v=((x_v,y_v),(w_v,y_v))$ iff u is a maximal segment that covers v and the absolute difference between y_u and y_v is 1 unit of separation. A definition follows analogously for a vertical type I escape segment. In Figure 1.a, u is a horizontal type I escape segment and r is a vertical type I escape segment.

A horizontal segment $u=((x_u,y_u),(w_u,y_u))$ in the routing region is a *type II escape segment* iff u is a maximal segment that intersects terminals a or b. A definition follows analogously for a vertical type II escape segment in the routing region. Note, the terminals a and b are considered to be in the routing region. In Figure 1.a, c is a horizontal type II escape segment that intersects terminal a, and d is a vertical type II escape segment that intersects terminal b.

A segment is an *escape segment* iff it is either a type I or type II escape segment. In Figure 1.b, we give a problem instance and its set of escape segments.

A horizontal segment $s=((x_s,y),(w_s,y))$ is an *escape subsegment* for horizontal escape segment $u=((x_u,y),(w_u,y))$ iff $x_u \leq x_s$ and $w_s \leq w_u$. A definition follows analogously for an escape subsegment of a vertical escape segment.

The point (x,y) is an *intersection point* iff there exists escape segments u and v such that (x,y) lies on both u and v.

A segment is *degenerate* iff the line segment consists of a single point.

LIMITING THE SEARCH AREA FOR A SINGLE LAYER CIRCUIT MEDIUM

By exploiting the theorem presented below, a router can restrict its search for a single layer optimal wiring to wires that only use escape subsegments as routing segments.

Theorem 1: If a wiring exists for terminals a and b on a single layer circuit medium then there is an optimal wiring on the single layer for a and b with respect to bend minimization that only makes use of escape subsegments and there is an optimal wiring on the single layer for a and b with respect to length minimization that only makes use of escape subsegments.

Proof: We will prove Theorem 1 by transforming an arbitrary wiring that is optimal with respect to either bend or length minimization into an optimal solution composed only of escape subsegments.

Let $W=(r_1,\cdots r_n)$ be a wire that connects a and b optimally with respect to either bend or length minimization. Without loss of generality, we assume each pair (r_i,r_{i+1}) forms a bend, $1 \leq i < n$. If this is not the case for some r_i and r_{i+1} then r_i and r_{i+1} can be combined into a larger routing segment without loss of optimality, $1 \leq i < n$. If W is composed strictly of escape subsegments then we have shown Theorem 1. Therefore, we assume there is at least one routing segment in W that is not an escape subsegment.

Let r_i be the minimally indexed routing segment of W that is not an escape subsegment. As there is a type II escape segment intersecting a for each possible direction into the routing region, and as there is a type II escape segment intersecting b for each possible direction into the routing region, r_1 and r_n are escape subsegments. As r_1 and r_n are escape subsegments, $1 < i < n$.

Our transformation of W will replace r_i with an equally long escape subsegment and will make modifications to r_{i-1} and r_{i+1}. The result of the transformation will be an optimal solution $\hat{W}$ with one less non-escape subsegment in it than in optimal solution W. We will show that these replacements and modifications can be made in isolation. Thus, we will be able to sequentially transform an arbitrary optimal solution to an optimal solution using only escape subsegments. There are two cases we must consider with respect to the r_i's direction: r_i is horizontal and r_i is vertical. As the two cases are analogous, we only present the case where r_i is horizontal.

Let $r_i=((x,y),(w,y))$. Let $u=((x_u,y_u),(w_u,y_u))$ be the horizontal boundary segment that r_i covers with minimal y-coordinate value greater than y. Let $v=((x_v,y_v),(w_v,y_v))$ be the

horizontal boundary segment that r_i covers with maximal y-coordinate value less than y. Let $\Delta_u = y_u - y$. Let $\Delta_v = y - y_v$ (Figure 2.a).

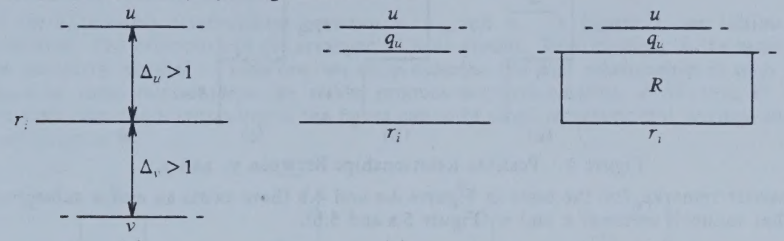


Figure 2 – u, r_i, q_u, and R Relationships

As r_i is not an escape subsegment, both Δ_u and Δ_v are greater than 1. As $\Delta_u > 1$ and u is a boundary segment, u has an escape segment, q_u, that lies between u and r_i (Figure 2.b).

Let R be the rectangular subregion of the circuit medium bounded by the points $(x, y_u - 1)$, $(w, y_u - 1)$, (x, y), and (w, y) (Figure 2.c). As r_i would cover any boundary segment in R, and as R is below u, and as u is the lowest boundary segment above r_i that r_i covers, there are no boundary segments in R. Therefore, R is a subregion of the routing region.

Let S be the set of all horizontal escape segments that intersect region R. Note, all escape segments in S are below u (y-coordinate of these escape segments are less than y_u). Furthermore, q_u is an element of S. Also, as escape segments have maximal length, any horizontal escape segment in S spans across from column x to column w.

Let $c = (x, y_c)$ be the upper endpoint of r_{i-1}. Let $d = (w, y_d)$ be the upper endpoint of r_{i+1}. Let $g = (w, y_g)$ be the lower endpoint of r_{i+1}.

There are eight cases for r_i we must consider with respect to the relative positions of r_{i-1} and r_{i+1} (Figure 3). The four cases in Figures 3.a – 3.d have analogous proofs. Similarly, the four cases in Figures 3.e – 3.h have analogous proofs. Therefore, we only present the proofs for Figures 3.a and 3.e. We call Figure 3.a and 3.e respectively the *stair* and *horseshoe* cases.

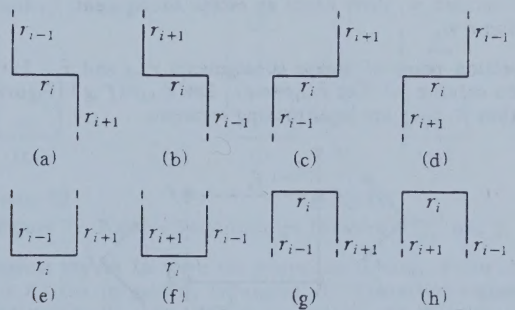


Figure 3 – Eight Relative Positions of Routing Segment r_i

Stair Case Transformation

There are 3 possible relationships between y_u and y_c in the stair case: $y_u < y_c$, $y_u = y_c$, and $y_u > y_c$ (Figure 4).

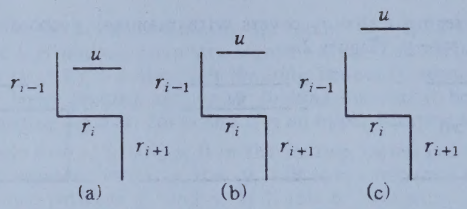


Figure 4 – Possible Relationships Between y_u and y_c

From earlier remarks, for the cases in Figures 4.a and 4.b there exists an escape subsegment $\hat{r}_i$ of q_u that connects columns x and w (Figure 5.a and 5.b).

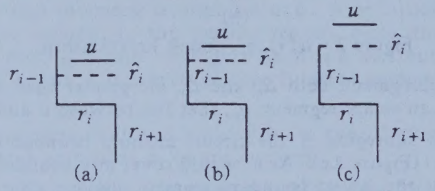


Figure 5 – Possible Positions for $\hat{r}_i$

The below remarks demonstrate for the case in Figure 4.c that there exists an escape subsegment $\hat{r}_i$ that connects column x at point c to column w (Figure 5.c).

Assume $i=2$. As r_1 is an escape subsegment of a type II escape segment and as the lower endpoint of r_1 connects to r_2, $c=a$. As $c=a$ and as $c \in R$, there exists an escape segment (type II), q_c, in S that traverses across R from point c to column w.

Assume $i>2$. As r_{i-2} and r_{i-1} are escape subsegments that connect at point c, c is an intersection point. Let q_c be the escape segment of horizontal escape subsegment r_{i-2}. As R is a subregion of the routing region and as escape segment q_c intersects R at c, q_c traverses across R from point c to column w.

As $i>1$ and as whether $i=2$ or $i>2$ there exists an escape segment q_c that traverses R from point c to column w, there exists an escape subsegment, $\hat{r}_i$, that connects column x at point c to column w.

Let e be the intersection point of escape subsegments r_{i-1} and $\hat{r}_i$. Let f be the point of intersection of $\hat{r}_i$ and column w. Let $\hat{r}_{i-1}=(c,e)$. Let $\hat{r}_{i+1}=(f,g)$ (Figure 6). As $\hat{r}_{i-1}$, $\hat{r}_i$, and $\hat{r}_{i+1}$ are entirely within R, they are legal routing segments.

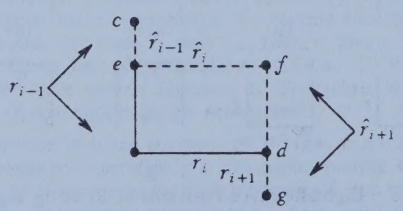


Figure 6

Transforming W into $\hat{W}$ by replacing r_{i-1}, r_i, r_{i+1} with $\hat{r}_{i-1}, \hat{r}_i, \hat{r}_{i+1}$, neither introduces extra bends nor increases the wire length. Thus, $\hat{W}$ is optimal and there is one less non-escape subsegment in it than in W. As the transformation is independent of any prior transformations, the transformation can be repeatedly applied. Therefore, we can successfully transform any stair case.

We note, for the stair case of Figure 5.c there is one less bend as c and e are the same point. As W is optimal, the case of Figure 5.c does not occur in a minimal bend solution.

In transforming W into $\hat{W}$, none of $\hat{r}_{i-1}, \hat{r}_i$, or $\hat{r}_{i+1}$ intersects any other routing segment r_j to form a cycle. If this situation did arise, the cycle could then be removed from $\hat{W}$ and thus producing a solution with less length and less bends than in optimal solution W. These

remarks also apply to the transformations given below for the horse shoe case.

Horseshoe Case Transformation

There are 13 possible relationships between y, y_u, and y_c. In Figure 7, we exhibit these relationships. The relationships are arranged in four groups. Relationships in the same group can be similarly handled. Therefore, we only examine the first relationship in each group. For each of these relationships, we either produce a transformation of W into $\hat{W}$ or we demonstrate that the relationship in the figure can only occur in suboptimal wirings and thus does not occur in W.

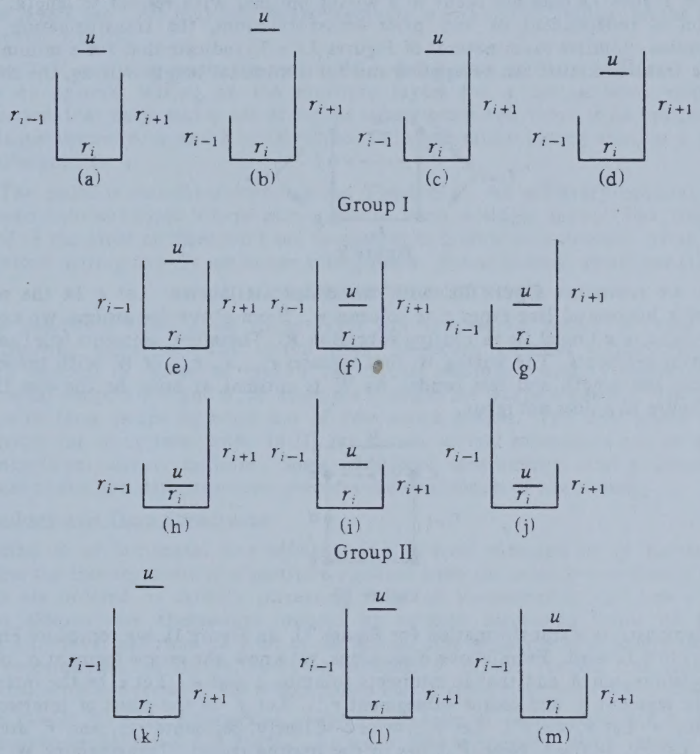


Figure 7 – Possible Relationships Between y, y_u, and y_c

In Figure 8, we reproduce Figure 7.a with the endpoints labeled. From above discussions, we know the segment (c,d) lies in routing subregion R. Therefore, segment (c,d) is a valid routing segment. The wiring $\hat{W}$ that replaces r_{i-1}, r_i, r_{i+1} of W with (c,d) has less length and less bends. As W is optimal, it must be the case that the situation in Figure 7.a does not occur. Similar examinations of Figures 7.b – 7.d demonstrate that these situations do not occur in an optimal wiring. As Figures 7.a – 7.d do not occur in an optimal wiring, no transformation is necessary.

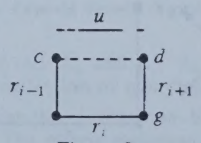


Figure 8

In Figure 9. we reproduce Figure 7.e with the endpoints labeled. From above discussions. we know the escape segment q_u of u lies in the routing subregion R and that it intersects columns x and w. Let e be the intersection point of escape segment q_u and escape subsegment r_{i-1}. Let f be the point of intersection of q_u and column w. Let $\hat{r}_i=(e,f)$. Let $\hat{r}_{i-1}=(c,e)$. Clearly, segments $\hat{r}_{i-1}$ and $\hat{r}_i$ are escape subsegments. Let $\hat{r}_{i+1}=(f,g)$. Note, $\hat{r}_{i+1}$ is in the routing region. Transforming W into $\hat{W}$ by replacing r_{i-1}, r_i, r_{i+1} with $\hat{r}_{i-1}$, $\hat{r}_i$, $\hat{r}_{i+1}$, neither introduces extra bends nor increases the wire length. Thus, $\hat{W}$ is optimal and there is one less non-escape subsegment in it than in W. We note that the transformation decreases the length. Hence, Figure 7.e does not occur in a wiring optimal with respect to length. As the transformation is independent of any prior transformations, the transformation can be repeatedly applied. Similar examinations of Figures 7.f – 7.j indicate that for a minimal bend wiring, a like transformation can be applied and for a minimal length wiring, the situations cannot occur.

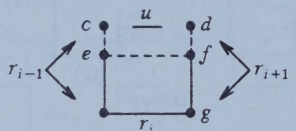


Figure 9

In Figure 10, we reproduce Figure 7.k with the endpoints labeled. Let e be the point of intersection of a horizontal line from c to column w. From above discussions, we know the segments (c,e) and (e,d) must lie in routing subregion R. Therefore, segments (c,e) and (e,d) are valid routing segments. The wiring $\hat{W}$ that replaces r_{i-1}, r_i, r_{i+1} of W with the sequence (c,d), (d,e) has less length and less bends. As W is optimal, it must be the case that the situation in Figure 10.k does not occur.

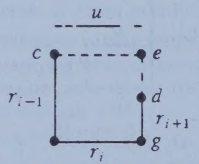


Figure 10

Finally, we demonstrate a transformation for Figure 7.l. In Figure 11, we reproduce Figure 7.l with the endpoints labeled. From above discussions, we know the escape segment q_u of u lies in the routing subregion R and that it intersects columns x and w. Let e be the intersection point of escape segment q_u and escape subsegment r_{i-1}. Let f be the point of intersection of q_u and column w Let $\hat{r}_i=(e,f)$. Let $\hat{r}_{i-1}=(c,e)$. Clearly, segments $\hat{r}_{i-1}$ and $\hat{r}_i$ are escape subsegments. Let $\hat{r}_{i+1}=(f,g)$. Note, $\hat{r}_{i+1}$ lies in the routing region. Transforming W into $\hat{W}$ by replacing r_{i-1}, r_i, r_{i+1} with $\hat{r}_{i-1}$, $\hat{r}_i$, $\hat{r}_{i+1}$, neither introduces extra bends nor increases the wire length. In fact, the transformation decreases the length. Hence, Figure 7.l does not occur in a wiring optimal with respect to length. However, $\hat{W}$ is optimal with respect to bend minimization and there is one less non-escape subsegment in it than in W. As the transformation is independent of any prior transformations, the transformation can be repeatedly applied. A similar examination of Figure 7.m indicates that for a minimal bend wiring, a like transformation can be applied and for a minimal length wiring, the situation cannot occur.

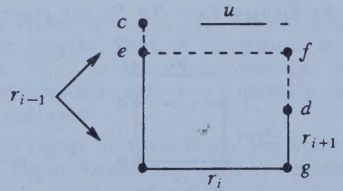


Figure 11

As for any horseshoe and stair case that can occur in W we can successfully apply a transformation, and as the transformations need be applied at most $n-2$ times to produce an optimal solution composed strictly of escape subsegments, Theorem 1 has been shown. □

LIMITING THE SEARCH AREA FOR A MULTILAYER CIRCUIT MEDIUM

For a multilayer circuit medium, we extend our definition of type II escape segments to include vias as well as terminals. We also extend our input specifications – there is an indicator on which layers a terminal, via, or boundary segment occurs. There is similar indicator of the layer associated with each escape segment. Thus, an object that appears on multiple layers can have multiple escape segments where the individual escape segments can have varying lengths.

Theorem 2: If a wiring exists for terminals a and b on a multilayer circuit medium then there is an optimal wiring on the multiple layers for a and b with respect to bend minimization that only makes use of escape subsegments and there is an optimal wiring on the multiple layers for a and b with respect to length minimization that only makes use of escape subsegments.

Proof: The proof is straight–forward given Theorem 1. An arbitrary optimal wire can be decomposed into subwires where each subwire is on a single layer. The transformations presented in the proof of Theorem 1 can be applied to individual subwires. Thus, it is possible in an optimal wiring to only use escape subsegments. For additional details see [1].

A LINE INTERSECTION ROUTER

We now describe a three–phase single–layer line intersection router, *lir*, that exploits the search limiting theorems. The initial phase is escape segment construction. For each boundary segment in the set of module and circuit medium boundary segments M and the two terminal endpoints a and b, *lir* attempts to construct escape segments. The middle phase creates with these escape segments one of two search graphs. The final phase examines the search graph for an optimal path. In [1], we discuss several extensions, one of which allows for a multilayer circuit medium. Some additional terminology, and a discussion of the individual phases and data structures precedes the description of the router.

Terminology and Data Structures

A partition H of horizontal line segments is a *bi–level* partition iff H has the following properties: the line segments in a partition element have the same y–coordinate; the partition elements are ordered by strictly increasing value of y–coordinate, and line segments in a partition element are themselves ordered by strictly increasing value of x–coordinate. Clearly, a bi–level partition of a set of horizontal line segments is uniquely determined by the above properties. A similar bi–level partition V of the vertical line segments can be constructed with a reversal of the x–coordinate and y–coordinate roles in the definition.

A graph $G=(N,E)$ is an *intersection graph* for the set of escape segments S iff G has the following properties: $N=S$ and $E=\{(u,v) \mid u,v \in N$, line segments u and v intersect$\}$.

A graph $G=(N,E)$ is an *escape graph* for the set of escape segments S iff G has the following properties: $N=A \bigcup B$ (A is the set endpoints for the set S, and B is the set of intersection points) and $E=\{(u,v) \mid u,v \in N$, u and v lie on the same escape segment in $S\}$.

Escape Segment Construction

The initial phase creates the type I escape segments by constructing the horizontal bi–level partition, H, and the vertical bi–level partition, V, from M and then determining the escape segments (if any) for each boundary segment in H and V. For a given boundary segment, the two limiting boundary segments for its escape segment are determined. From these two limiting boundary segments, the escape segment is created in a straight–forward fashion. For the two terminal endpoints a and b, type II escape segments that intersect the terminals are also determined.

The outputs of the first phase are two sets, L_h and L_v. L_h is the collection of non–degenerate horizontal escape segments; L_v is the collection of non–degenerate vertical escape segments.

The bi–level partitions H and V can be constructed by sorting M in $O(m \cdot \log_2 m)$ time, where m is the cardinality of M. The escape segments can then be constructed through a minor modification of Shamos' and Hoey's intersection point algorithm [8] in $O(m^2)$ time.

Graph Construction

The middle phase constructs from L_h and L_v either an intersection graph or an escape graph. An intersection graph is constructed if the optimization criterion is bend minimization; an escape graph is constructed if the optimization criterion is length minimization.

There is a node in the intersection graph for each escape segment constructed in the initial phase. In addition to these nodes, there are two special nodes, s and t, that are respectively the *source* and *sink* of the intersection graph. There is an edge from s to each of the escape segments of a and also an edge from t to each of the escape segments of b. The remaining edges of the intersection graph connect nodes whose corresponding escape segments intersect. In the intersection graph, node s is the degenerate line segment a and node t is the degenerate line segment b.

The intersection graph can be readily constructed from L_h and L_v by first transforming L_h and L_v into bi-level partitions and then applying a second minor modification of the Shamos' and Hoey's intersection point algorithm. The bi-level partition can be constructed in $O(m \log_2 m)$ time. The run-time of the intersection point algorithm is proportional to the number of edges in the intersection graph. As the number of nodes in the graph is $O(m)$ (at most one node for each boundary segment in M), the number of edges is $O(m^2)$. Hence, intersection graph construction is $O(m^2)$.

The construction of an escape graph is similar to the construction of and intersection graph - transforming L_h and L_v into bi-level partitions and then applying a modified intersection point algorithm. However, rather than determining whether two escape segments intersect, an escape graph indicates the horizontal and vertical escape subsegments associated with the intersection points. The source s and sink t of the escape graph are respectively a and b.

The number of nodes in the escape graph is $O(m^2)$ since in the worst case each boundary segment can intersect all other boundary segments. The number of edges is also $O(m^2)$ since each nodes has at most 4 edges (one for each rectilinear direction). Thus, escape graph construction is also $O(m^2)$.

Path Construction

The optimization criterion directs which one of two search techniques is to be invoked from root s of the graph $G=(N,E)$ produced in the middle phase. A breadth-first search of the intersection graph produces minimal bend wirings; a shortest-path search of the escape graph produces minimal length wirings.

The search of the intersection graph terminates either upon reaching node t or when there are no more unexplored nodes. If the latter condition causes termination then it is the case that there is no wiring from a to b. A property of a breadth-first search path from s to t is that it minimizes the number of edges in the path. As each edge in a breadth-first search path after the initial edge introduces a bend, a breadth-first search path from s to t is a minimal bend wiring with respect to the edges in the intersection graph. Clearly, such a path corresponds to a minimal bend wiring with respect to the escape subsegments and by applying Theorem 1 we conclude that the wiring is optimal.

A breadth-first search examination of a graph requires time proportional to the number of edges. As there are $O(m^2)$ edges in the intersection graph, we conclude that searching an intersection graph for an optimal wiring is $O(m^2)$.

An examination of the specification of an escape graph shows that for the single layer circuit medium that the graph is planar. A shortest path in a planar graph with $O(m^2)$ nodes can be determined in $O(m^2 \cdot \sqrt{\log_2 m})$ time [2]. Clearly, a shortest path in an escape graph corresponds to a minimal length wiring with respect to escape subsegments and by applying Theorem 1 we conculde that the wiring is optimal.

Correctness and Complexity

The correctness of *lir* follows immediately from Theorem 1 and from the above discussions.

For bend minimization, each phase requires $O(m^2)$ time. Hence, for bend minimization, *lir* is $O(m^2)$. For length minimization, the first two phases are also $O(m^2)$. However, the shortest path search is $O(m^2 \cdot \sqrt{\log_2 m})$. Hence, for length minimization, *lir* is $O(m^2 \cdot \sqrt{\log_2 m})$.

The largest data structure that must be maintained by *lir* is the escape graph which can have $O(m^2)$ nodes and $O(m^2)$ edges. Therefore, the space complexity of *lir* is $O(m^2)$. These time

and space complexities of *lir* compare favorably to existing methods.

Maze routers, which also guarantee optimal solutions, have a worst case time complexity of $O(r)$, where r is the number of points in the circuit medium. As the time spent on an line segment construction for line search routers in the worst case is a function of the length of the escape segment, these two routers are also $O(r)$.

Note, parameter r for some instances may be arbitrarily larger than m.

SUMMARY

We have proven that it is sufficient to only consider escape subsegments in looking for optimal wirings. We then briefly described a router *lir* that exploits these theorems to quickly produce optimal solutions with efficient space usage under either the bend or the length minimization criterion.

Algorithm *lir* operates by constructing a set of escape segments that intersect the terminal endpoints, surround the modules, and enclose the interior of the circuit medium. With the escape segments, *lir* creates either an intersection or escape graph. A breadth-first search of the intersection graph yields a minimal bend wiring; a shortest-path search of the escape graph yields a minimal length wiring. Its worst-case run-time complexity $O(m^2 \cdot \sqrt{\log_2 m})$ is a significant improvement over other general purpose routers. As its complexity is independent of the length of the wiring, for some problem instances this independence introduces an arbitrary speed-up.

REFERENCES

1. J. P. Cohoon, A Line Intersection Algorithm for the Routing Problem, DAMACS 84-02, University of Virginia, 1984.
2. G. Frederickson, Shortest Path Problems in Planar Graphs, *24th Annual Symposium on Foundations of Computer Science*, 1983, 242-247.
3. W. Heyns, W. Sansen and H. Beke, A Line-Expansion Algorithm for the General Routing Problem with a Guaranteed Solution, *17th Design Automation Conference Proceedings*, 1980, 243-249.
4. D. W. Hightower, A Solution to the Line-Routing Problem on the Continuous Plane, *6th Design Automation Workshop Proceedings*, 1969, 1-24.
5. D. W. Hightower, The Interconnection Problem: a Tutorial, *Computer*, *7(4)*, April 1974, 18-32.
6. C. Y. Lee, An Algorithm for Path Connections and Its Applications, *IRE Transactions on Electronic Computers*, *EC-10(3)*, September 1961, 346-365.
7. F. Rubin, The Lee Path Connection Algorithm, *IEEE Transactions on Computers*, *C-23(9)*, September 1974, 907-914.
8. M. I. Shamos and D. Hoey, Geometric Intersection Problems, *17th Annual Symposium on Foundations of Computer Science*, 1976.
9. J. Soukup, Circuit Layout, *Proceedings of the IEEE*, *69(10)*, October 1981, 1281-1304.

A 1.6 Approximation Algorithm for Routing Multiterminal Nets Around a Rectangle

Teofilo F. Gonzalez
Department of Computer Science
The University of California
Santa Barbara, CA 93106

Sing-Ling Lee
Programs in Computer Science
The University of Texas at Dallas
Richardson, Texas 75080

ABSTRACT

The problem of connecting a set of terminals that lie on the sides of a rectangle to minimize the total area is discussed. We present an $O(n^2)$ approximation algorithm to solve this problem. Our algorithm generates a solution with area $\leq 1.6 * OPT$, where OPT is the area of an optimal solution. The nets are routed according to the following greedy strategy: the wire connecting all points from a net is one that crosses the least number of corners. For some nets there are several paths that cross the least number of corners. A subset of these nets is connected by paths that blend with the connecting paths for other nets. The path for the remaining nets is selected using several strategies and 2^6 layouts are obtained. The best of these layouts is the solution generated by our algorithm.

I. INTRODUCTION

Let G be a rectangular component of size h by w (height by width). There are n terminals $(T_1, T_2, \ldots, T_n)$ on the sides of G. These terminals are partitioned into m subsets denoted by $N_1, N_2, \ldots, N_m$. Each subset N_i is called a net and it is assumed that $|N_i| > 1$ for all i. The problem depicted in figure 1.1 consists of five nets: $N_1 = \{T_2, T_4, T_7\}$, $N_2 = \{T_1, T_3, T_{10}, T_{14}\}$, $N_3 = \{T_5, T_{12}\}$, $N_4 = \{T_6, T_8, T_9\}$ and $N_5 = \{T_{11}, T_{13}, T_{15}\}$. It is assumed that every pair of terminals is at least $\lambda > 0$ units apart and every terminal is located at least λ units from each of the corners of G. All the terminals in each net must be made electrically common by connecting them with wires. The path followed by these wires can be partitioned into a finite number of straight line segments. Each of these line segments must lie on the same plane as G, be on the outside of G and be parallel to a side of G. Perpendicular line segments can intersect at any point, but parallel line segments must be at least λ units apart. Also, all line segments must be at least λ units away from every side of rectangle G except in the vicinity where a line segment connects a terminal. The *R1M problem* consists of specifying paths for all the interconnections subject to the rules mentioned above in such a way that the total area is minimized, i.e., place the component together with all the wires inside a rectangle (with the same orientation as G) of least possible area. This problem has applications in the layout of integrated circuits ([L] and [R]) and conforms to a set of design rules for VLSI systems [MC].

The 2-R1M is defined similarly, except that all nets are restricted to be of size two. Hashimoto and Stevens present an $O(n \log n)$ algorithm to solve the R1M problem for the case when all the points in S lie on one side of G. An $\Omega(n \log n)$ lower bound on the worst case time complexity for this problem was established in [GLL]. Algorithms to solve the 2-R1M problem appear in [La] and [GL2]. If more than two layers are allowed and wire overlap is permitted, the problem becomes an NP-hard problem [SBR]. Other generalizations of the 2-R1M problem have been shown to be NP-hard [La]. Gonzalez and Lee [GL3] present an $O(n^2)$ approximation algorithm for the R1M problem that generates a solution with area $\leq 1.69 * OPT$. In this paper we present an approximation algorithm with a worst case approximation bound of 1.6.

For i=2,3,4, let R1M-i denote a R1M problem in which all global nets contain terminals on exactly i sides of G. In section II we introduce our notation and present some basic results. In order to simplify the exposition of our results, we begin by presenting approximation algorithms for restricted versions of the R1M problem. In section III-V we present an approximation algorithms for the R1M-3, R1M4 and R1M-2 problems. In section VI we combine the results obtained in sections II-V to obtain our

1.6 approximation algorithm for the R1M problem.

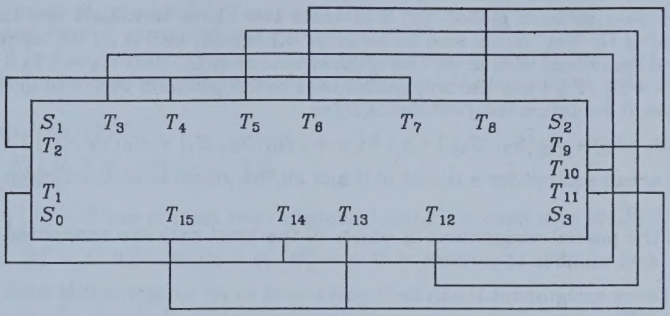


figure 1.1

II. DEFINITIONS AND BASIC RESULTS.

Label the sides of the component (in the obvious way) left, top, right and bottom. Starting in the bottom-left corner of G, traverse the sides of the rectangle clockwise. The ith corner to be visited is labeled S_{i-1}. Assume that the ith terminal visited is terminal T_i . The *close interval* $[x , y]$, where x and y are the corners of G or the terminals T_i, consists of all the points on the sides of G that are visited while traversing the sides of G in the clockwise direction starting at point x and ending at point y. Note that the interval [x , x] consists of only one point. Parentheses are used instead of square brackets for open intervals. We use $[S_0 , S_1]$, $[S_1 , S_2]$, $[S_2 , S_3]$ and $[S_3 , S_0]$; 0, 1, 2 and 3; L, T, R, and B; and L', T', R' and B' to represent the left, top, right and bottom sides of G, respectively. Terminal T_i is said to belong to side l, $S(i) = l$, if T_i is located in $[S_l , S_{(l+1) mod (4)}]$. The function I(j) indicates the index of the net to which terminal T_j belongs. The function L(j) is defined in such a way that the interval $[T_j , T_{L(j)}]$ is the smallest interval that includes all the terminals from net $N_{I(j)}$. Set D $= \{d_1 , d_2 , ..., d_k\}$ is said to be an *assignment* if the cardinality of set D is m and D contains exactly one index of a terminal from each net. Any subset of an assignment is said to be a *partial assignment*. An assignment D indicates the starting point for the path connecting all the points in each net. If i ∈ D then the path connecting all the terminals in net $N_{I(i)}$ starts at terminal T_i moving perpendicular to side S(i) and then it continues in the clockwise direction with respect to G until it reaches $T_{L(i)}$. Each terminal, T_k, in net $N_{I(i)}$ is joined to this path by a line segment perpendicular to side $S(T_k)$. The starting point for some connecting paths might not be defined in a partial assignment. The assignment for the layout given by figure 1.1 is {2 , 3 , 5 , 6 , 11}. For any l ∈ D, we say that the path connecting net $N_{I(l)}$ given by D *crosses* point z if z ∈ $[T_l , T_{L(l)}]$. If i ∈ D, we say that the path connecting net $N_{I(i)}$ *begins* at point T_i and *ends* at point $T_{L(i)}$.

For any assignment (or partial assignment) D we define the *height function* H_D for x,y ∈ $\{T_1 , T_2 , ..., T_n\} \cup \{S_0 , S_1 , S_2 , S_3\}$ as follows:

$$H_D(x , y) = \max\{ \text{number of paths given by D that cross point z} \mid z \in [x , y] \}.$$

For example, $H_D(S_0 , S_1)$ is 1, $H_D(T_5 , T_5)$ is 3 and $H_D(S_2 , S_3)$ is 3 for the assignment, D, whose layout appears in figure 1.1.

Lemma 2.1: For every assignment D, there is a rectangle Q of size h_Q by w_Q, where

$$h_Q = h + (H_D(S_1 , S_2) + H_D(S_3 , S_0)) * \lambda, \text{ and}$$
$$w_Q = w + (H_D(S_0 , S_1) + H_D(S_2 , S_3)) * \lambda$$

with the property that rectangle G together with the interconnecting paths defined by D can be made to fit inside Q. A layout with area $h_Q * w_Q$ can be constructed in O(n log n) time.

Proof: The proof is a direct generalization of the proof for the 2-R1M problem that appears in [L]. The algorithm that constructs the final layout uses as a subalgorithm

the procedure given in [GLL] and [HS].

Net N_i is said to be a *global net* if at least two of its terminals are located on opposite sides of G. Net N_i is said to be *local* otherwise, i.e., if all its terminals are located on the same side of G or on two adjacent sides of G. Nets $N_2 = \{ T_1 , T_3 , T_{10} , T_{14} \}$ and $N_3 = \{ T_5 , T_{12} \}$ are the only global nets in the problem depicted in figure 1.1. For assignment D we define the *function A(D)* as

$$(h + (H_D(S_1 , S_2) + H_D(S_3 , S_0)) * \lambda) * (w + (H_D(S_0 , S_1) + H_D(S_2 , S_3)) * \lambda)$$

i.e., the total area required for a layout of G and all the interconnections given by D.

Definition 2.1: D'

Let D' be the partial assignment in which all the local nets are connected by paths crossing the least number of corners of G.

Lemma 2.2: Every assignment D can be transformed to an assignment M such that $D' \subseteq$ M and $A(M) \leq A(D)$.

Proof: The proof follows the same lines as the one for the 2-R1M problem that appears in [L].

At this point our algorithm abandons the procedures given in [La] and [GL2]. The main difficulty that we encounter in extending the results for the 2-R1M problem to this problem is that the divide and conquer step seems not applicable. The reason for this is that there seems to be no rule to separate the nets with terminals located in three or four sides of G into groups that can be routed independently of each other.

III. APPROXIMATION ALGORITHM FOR THE R1M-3 PROBLEM.

In this section we present an approximation algorithm for the R1M-3 problem. Let M_3 be the set of global nets with terminals located on exactly three sides of G. Clearly, all global nets belong to set M_3. For the set of nets M_3 we construct assignment D_3 as follows: $D_3 = \{$ all nets in M_3 are connected by paths that cross the least number of corners of G$\}$. Let M_3^{TB} (M_3^{LR}) be the set of all nets in M_3 without terminals located on either the left or right (top or bottom) side of G. Let m_3^{TB} and m_3^{LR} represent the number of elements in M_3^{TB} and M_3^{LR} respectively.

Lemma 3.1: Let D be an optimal assignment such that $D' \subseteq$ D. Let M be D except that all nets in $M_3^{TB} \cup M_3^{LR}$ which are assigned as in our algorithm. Then

$$H_M(S_1,S_2) + H_M(S_3,S_0) \leq H_D(S_1,S_2) + H_D(S_3,S_0) + x_3^{TB}, \text{ and}$$

$$H_M(S_0,S_1) + H_M(S_2,S_3) \leq H_D(S_0,S_1) + H_D(S_2,S_3) + x_3^{LR},$$

where x_3^{TB} (x_3^{LR}) is the number of nets in X_3^{TB} (X_3^{LR}) and X_3^{TB} (X_3^{LR}) contains all the nets in set M_3^{TB} (M_3^{LR}) that are connected differently in D and M.

set	contribution to our lower bound for $h/(\lambda)+H_D(S_1,S_2)+H_D(S_3,S_0)$	contribution to our lower bound for $w/(\lambda)+H_D(S_0,S_1)+H_D(S_2,S_3)$
M_3^{TB}	$1.5m_3^{TB} + 0.5x_3^{TB}$	$2m_3^{TB} + 0.5x_3^{TB}$
M_3^{LR}	$2m_3^{LR} + 0.5x_3^{LR}$	$1.5m_3^{LR} + 0.5x_3^{LR}$

Table 3.1: Lower bounds for the R1M3 problem.

Lemma 3.2: Let D be an optimal assignment such that $D' \subseteq$ D. Then assignment D and rectangle G satisfy the lower bounds given in table 3.1.

Theorem 3.1: For the R1M-3 problem, let D be an optimal assignment such that $D' \subseteq$ D and let M be the assignment generated by our algorithm. Then, $A(M) \leq 1.6 * A(D)$.

IV. APPROXIMATION ALGORITHM FOR THE R1M-4 PROBLEM.

In this section we present an approximation algorithm for the R1M-4 problem. Let M_4 be the set of all global nets with terminals located on the four sides of G and let m_4 represent the number of nets in it. Clearly, all global nets belong to set M_4. Nets N_i and N_j (both in set M_4) are said to be *agreeable* if on some side of G no terminal from net N_i is between any two terminals from net N_j and no terminal from net N_j is between any two terminals from net N_i. Procedure PARTITION defined below partitions the set M_4 into the sets M_4^A, M_4^N, M_4^T, M_4^R, M_4^B, M_4^L, M_4^T, M_4^R and M_4^B. Since some of

these sets consist of 1-tuples, 2-tuples and 4-tuples, by partition we mean that every element in set M_4 is in one and only one of the tuples in these sets. After the algorithm terminates, let m_4^X represent the number of tuples in M_4^X, for X ∈ { A , N , T' , R' , B' , L' , T , R , B }. It is very important to keep in mind that sets M_4^A , M_4^T , $M_4^{R'}$, $M_4^{B'}$ and $M_4^{L'}$ consists of 1-tuples, set M_4^N has only 4-tuples and the remaining sets contain only 2-tuples.

procedure PARTITION

$M_4^A \leftarrow \{ N_i \mid N_i \in M_4$ and $|N_i| = 4 \}$;
W ← M_4 - M_4^A;

Z ← { $N_i \mid N_i \in$ W has at least two terminals located on each side of G};
W ← W - Z;
$M_4^T \leftarrow \phi$; $M_4^R \leftarrow \phi$; $M_4^B \leftarrow \phi$; $M_4^L \leftarrow \phi$; $M_4^N \leftarrow \phi$;

while |Z| ≥ 4 *do*
 let N_i, N_j, N_k and N_l be any four nets in Z;
 if any two of these four nets are agreeable
  then delete from Z two agreeable nets (two out of the four) and add them as a pair to M_4^T if these two nets are agreeable on the left or right side of G, otherwise add these two nets as a pair to M_4^R
  else delete all four nets from Z
   $M_4^N \leftarrow M_4^N \cup \{ (N_i, N_j, N_k, N_l) \}$
 endif;
endwhile;

// later on we explain what to do when Z ≠ φ //

while there is a net in Z with at least two terminals located on side X and exactly one terminal located on the remaining sides of G *do*
 Let y be one of such nets and X be the side on which it has more than one terminal;
  Z ← Z - { y };
  $M_4^X \leftarrow M_4^X \cup \{ y \}$;
endwhile;

W ← W ∪ Z;
Z ← { $N_i \mid N_i \in$ W has at least two terminals located on the top side if G};
W ← W - Z;

while there are two nets in Z with exactly one terminal located on the bottom side of G *do*
 delete two of such nets from Z and add them as a pair to M_4^T;
endwhile;

while there are two nets in Z with exactly one terminal located on the same side of G *do*
 delete two of such nets from Z and add them as a pair to M_4^T
endwhile;

// later on we explain what to do when Z ≠ φ //

Z ← { $N_i \mid N_i \in$ W has at least two terminals located on the right side of G };
W ← W - Z;

while there are two nets in Z with exactly one terminal located on the left side of G *do*
 delete two of such nets from Z and add them as a pair to M_4^R;
endwhile;

while |Z| ≥ 2 *do*
 delete any two nets from Z and add them as a pair to M_4^R
endwhile;

// later on we explain what to do when Z ≠ φ //

Z ← { $N_i \mid N_i \in$ W has at least two terminals located on the bottom side of G };
W ← W - Z;

while |Z| ≥ 2 *do*
 remove any two nets from Z and add them to M_4^B as a pair;
endwhile;

// later on we explain what to do when $Z \neq \phi$ //

end of procedure

For some problem instances the algorithm fails to assign some nets to any of the sets M_4^X, $X \in \{ A, N, T', R', B', L', L, T, R, B \}$. Let us assume that this is not the case, i.e., Z is always empty whenever we reach each of the "// later on ... //" comments in our procedure. In section VI we explain what to do when this is not the case.

a) Construction of the Assignment D_A''.

In what follows we explain how the assignment D_A'' for all the nets in set M_4^A is to be constructed. Figure 4.1 gives a layout for the assignment D_A'' constructed by our procedure for a set M_4^A with four nets. For any permutation, π, of the nets in set M_4^A and an integer i ($0 \leq i \leq 3$) we define an assignment (denoted ASG(π,i)) as follows: the first net in π is connected by a path that begins on side i and ends on side (i+3)mod(4); and the path connecting the kth net in π begins on side j and ends on side (j+3)mod(4), where j is the side on which the path connecting the (k-1)st net in π ends. We claim that for every integer i ($0 \leq i \leq 3$) there is a permutation, π (that depends on i) of the nets in set M_4^A such that there is a layout for assignment ASG(π,i) with the property that for any k ($1 < k \leq m_4^A$) the path connecting the (k-1)st net in π and the path connecting the kth net in π can share the same track on the side where the path connecting the (k-1)st net ends. In this case we say that π *is a valid permutation with respect to i*. We claim that every integer i, $0 \leq i \leq 3$, there is a valid permutation for the set of nets in M_4^A. Assignment D_A'' is just ASG(π , 1).

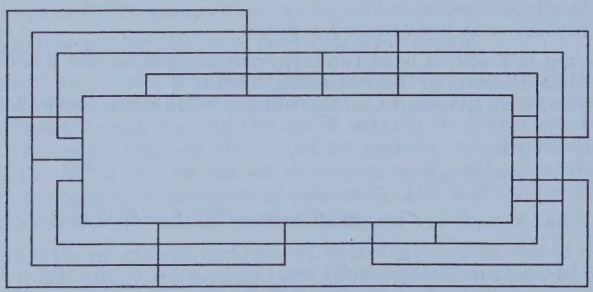


Figure 4.1: Assignment D_A'' for the case when $m_4^A = 4$.

b) Construction of the Assignment D_N''.

The assignment D_N'' is constructed by applying the following rule to each 4-tuple in M_4^N. Let (N_i, N_j, N_k , N_l) be any tuple in set M_4^N. Let us assume that these nets have been ordered in such a way that N_i is the net (amongst these four nets) whose bottommost terminal located on the left side of G is closest to the top side of G; net N_j is the net (amongst N_j, N_k and N_l) whose rightmost terminal located on the bottom side of G is closest to the left side of G; and net N_l is the net (amongst N_k and N_l) whose leftmost terminal located on the top side of G is closest to the right side of G. A layout for the assignment constructed for these nets is given in figure 4.2.

c) Construction of assignment D_X'' for $X \in \{ T', R', B', L' \}$.

Assignment D_X'', $X \in \{ T', R', B', L' \}$, is constructed by a procedure similar to the one used in part (a). For brevity we will not discuss it.

d) Construction of assignment D_X'' for $X \in \{ T, R, B \}$.

Assignment D_X'', $X \in \{ T, R, B \}$, for the set M_4^X is constructed by applying the following rule to each 2-tuple in M_4^X. Let (N_i , N_j) be any 2-tuple in set M_4^X. From algorithm PARTITION we know that both of these nets are agreeable on at least one side of G because both of these nets have exactly one pin located on some side of G. These nets are assigned in such a way that there is a layout in which these nets share the same track on side Z of G and both nets have exactly one terminal located on side Z. Side Z is selected using the following priorities: the opposite side of X has the highest priority; and the adjacent sides of G have the lowest priorities. Note that side Z cannot be the same as side X.

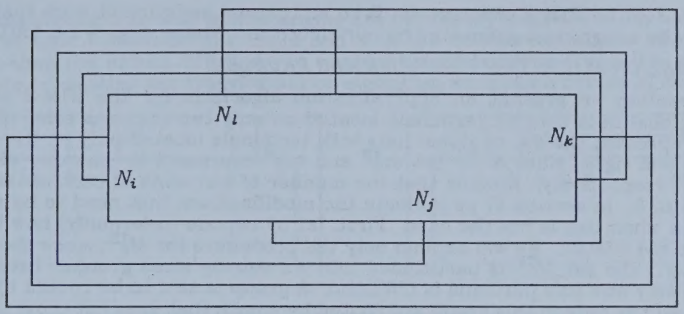


Figure 4.2: Assignment for a tuple in M_4^N.

e) Final Assignment and Proof of Approximation Bound.

Let $D_4 = D_A'' \cup D_N'' \cup D_{T'}'' \cup D_{R'}'' \cup D_{B'}'' \cup D_{L'}'' \cup D_T'' \cup D_R'' \cup D_B''$. In what follows we assume that m_4^A is a multiple of four. In section VI we indicate what to do when this is not the case.

Lemma 4.1: Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except that all the nets in $M_4^A \cup M_4^N$ are routed as in our approximation algorithm. Then

$H_M(S_1,S_2) + H_M(S_3,S_0) \leq H_D(S_1,S_2) + H_D(S_3,S_0) + 0.5m_4^A + 1 + 7x * M_4^N$, and

$H_M(S_2,S_3) + H_M(S_0,S_1) \leq H_D(S_2,S_3) + H_D(S_0,S_1) + 0.5m_4^A + 7y * M_4^N$, where x and y are nonnegative reals such that $x + y = 1$.

In what follows we assume that m_4^X, $X \in \{ T' , R' , B' , L' \}$, is a multiple of four. In section VI we indicate what to do when this is not the case.

Lemma 4.2: Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except for all the nets in M_4^X, $X \in \{ T' , R' , B' , L' \}$, are routed as in our approximation algorithm. Then

$$H_M(S_1,S_2) + H_M(S_3,S_0) \leq H_D(S_1,S_2) + H_D(S_3,S_0) + a * m_4^X, \text{ and}$$
$$H_M(S_2,S_3) + H_M(S_0,S_1) \leq H_D(S_2,S_3) + H_D(S_0,S_1) + b * m_4^X, \text{ where}$$
$$a = \begin{cases} 3 & \text{if } X \in \{T', B'\} \\ 2 & \textit{otherwise}; \end{cases} \quad \text{and} \quad b = \begin{cases} 2 & \text{if } X \in \{T', B'\} \\ 3 & \textit{otherwise}. \end{cases}$$

For $X \in \{ T , R , B \}$ let p_4^X be the fraction of pairs in M_4^X that are connected in D_X'' by paths that do not overlap on the top or bottom side of G, and $q_4^X = 1 - p_4^X$. Since all pairs of nets in M_4^B are agreeable on the top side of G, we know that $q_4^B = 0$.

Lemma 4.3: Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except that all nets in M_4^X ,$X \in \{ T, R, B \}$, are routed as in our approximation algorithm. Then

$$H_M(S_1,S_2) + H_M(S_3,S_0) \leq H_D(S_1,S_2) + H_D(S_3,S_0) + p_4^X * m_4^X + 2q_4^X * m_4^X,$$
$$H_M(S_2,S_3) + H_M(S_0,S_1) \leq H_D(S_2,S_3) + H_D(S_0,S_1) + 2p_4^X * m_4^X + q_4^X * m_4^X.$$

set	contribution to our lower bound for $h/(\lambda)+H_D(S_1,S_2)+H_D(S_3,S_0)$	contribution to our lower bound for $w/(\lambda)+H_D(S_0,S_1)+H_D(S_2,S_3)$
M_4^A	$2.5m_4^A$	$2.5m_4^A$
M_4^N	$14m_4^N$	$14m_4^N$
$M_4^{T'} \cup M_4^{B'}$	$4m_4^{T'} + 4m_4^{B'}$	$4.5m_4^{T'} + 4.5m_4^{B'}$
$M_4^{R'} \cup M_4^{L'}$	$4.5m_4^{R'} + 4.5m_4^{L'}$	$4m_4^{R'} + 4m_4^{L'}$
$M_4^T \cup M_4^B$	$p_4^T * m_4^T + 5m_4^T + 6m_4^B$	$q_4^T * m_4^T + 6m_4^T + 6m_4^B$
M_4^R	$p_4^R * m_4^R + 6m_4^R$	$q_4^R * m_4^R + 5m_4^R$

Table 4.1: Lower bounds for the R1M-4 problem.

Lemma 4.4: Let D be an optimal assignment such that $D' \subseteq D$. Then assignment D and rectangle G satisfy the lower bounds given in table 4.1.

Theorem 4.1: For the R1M-4 problem, let D be an optimal assignment such that $D' \subseteq D$ and let M be the assignment generated by our algorithm. Then, $A(M) \leq 1.6 * A(D)$.

V. APPROXIMATION ALGORITHM FOR THE R1M-2 PROBLEM.

In this section we present an approximation algorithm for the R1M-2 problem. Clearly, all global nets contain terminals located on only two opposite sides of G. Let M_2^{TB} (M_2^{LR}) represent the set of global nets with terminals located only on the top and bottom (left and right) sides of G. Let m_2^{TB} and m_2^{LR} represent the number of nets in M_2^{TB} and M_2^{LR} respectively. Assume that the number of elements in each of these sets is a multiple of 6. In section VI we indicate the modifications that need to be made to the algorithm when this is not the case. First, let us explain (informally) how the nets in these sets are routed. We will explain only the procedure for M_2^{TB}, since the one for M_2^{LR} is similar. The set M_2^{TB} is partitioned into six equally sized groups. Later on we explain precisely how this partition is obtained. A group is said to be routed by a path type L (R) if all the nets in this group are routed by a path that does not cross the right (left) side of G. We will construct 2^6 assignments. Each of these assignments corresponds to a string of six elements from the alphabet { L , R }. The ith element in a string specifies the type of path used to connect all nets in the ith group. Let us now formally define this construction process. For each net N_j in M_2^{TB} (M_2^{LR}), let p(j) be the index of the leftmost (bottommost) terminal of N_j located on the top (left) side of G. Let $P^{TB} = \{ p(j) \mid N_j \in M_2^{TB} \}$, and $P^{LR} = \{ p(j) \mid N_j \in M_2^{LR} \}$. Each of these sets is partitioned into the sets N_i^{TB} and N_i^{LR} for $0 \leq i \leq 5$ as follows:

$$N_i^{TB} = \{ I(j) \mid j \text{ is the kth smallest value in the set } P^{TB} \text{ and } (i/5)*|P^{TB}| < k \leq ((i+1)/5)*|P^{TB}| \} \text{ and}$$

$$N_i^{LR} = \{ I(j) \mid j \text{ is the kth smallest value in the set } P^{LR} \text{ and } (i/5)*|P^{LR}| < k \leq ((i+1)/5)*|P^{LR}| \}.$$

We define the following sets for $0 \leq i \leq 63$:

D_i^{TB} = { if in the binary representation of i the $(l+1)$st least significant bit is one then the connecting path for each net in set N_l^{TB} does not cross corner S_2, otherwise the connecting path for such a net does not cross corner $S_1 \mid 0 \leq l \leq 5$}, and

D_i^{LR} = { if in the binary representation of i the $(l+1)$st least significant bit is one then the connecting path for each net in set N_l^{LR} does not cross corner S_1, otherwise the connecting path for such a net does not cross corner $S_0 \mid 0 \leq l \leq 5$}.

For $0 \leq i,j \leq 63$, define assignment $D_{i,j}$ as $D_{i,j} = D' \cup D_i^{TB} \cup D_j^{LR}$, and let P be one of the $D_{i,j}$'s of least area.

In lemma 5.1 we assume that when interchanging the connecting paths of two nets in M_2^{TB} that cross on the top side of G will increase by at most 2 the vertical height of the assignment. This assumption is not always true. In figure 5.1 we show one counterexample to this assertion. We call this interchange a *type I interchange*. The two nets involved in this interchange form a *type I pair*. We also assume that when interchanging the connecting path for a net in M_2^{TB} to one that does not cross the left or right sides of G, will increase the vertical height of an assignment by at most two. This statement always holds true.

Lemma 5.1: Let D be an optimal assignment such that $D' \subseteq D$. There is an assignment $D_{i,j}$ (constructed by our algorithm) such that if M is defined as D except for all the nets in $M_2^{TB} \cup M_2^{LR}$ which are routed as in the assignment $D_{i,j}$, then

a) $H_M(S_1,S_2) + H_M(S_3,S_0) \leq H_D(S_1,S_2) + H_D(S_3,S_0) + (0.6) * m_2^{TB}$

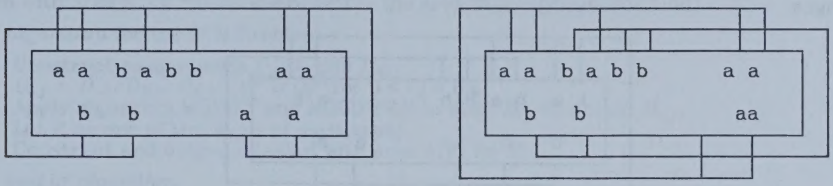
b) $H_M(S_0,S_1) + H_M(S_2,S_3) \leq H_D(S_0,S_1) + H_D(S_2,S_3) + (0.6) * m_2^{LR}$.

It is assumed that when D is transformed to any of the assignments $D_{i,j}$ and two paths that cross on the top side of G are interchanged, such an interchange is not a type I interchange.

Let L be an optimal solution that includes D'. The existence of at least one of these assignments is guaranteed by lemma 2.2. Let S be any of the $D_{i,j}$ assignments. The difference between S and L is the way in which some nets in $M_2^{TB} \cup M_2^{LR}$ are routed. Lemma 5.1 shows that at least one of our assignments differs in vertical height from L by at most $1.6 * m_2^{TB}$ and in horizontal height by at most $1.6 * m_2^{LR}$. As noted in the text appearing immediately before lemma 5.1, we cannot yet claim that the 1.6 bound holds when type I interchanges occur.

In order for our 1.6 bound to hold true we will transform each assignment S (each of the $D_{i,j}$'s) to another assignment R such that if the total increase in vertical height

when transforming L to S by interchanging the nets in M_2^{LR} was $\leq w + 3\alpha$, where α is the number of type I interchanges made and w is the contribution from all other interchanges, then the actual difference in vertical height between L and R is at most $w + 2\alpha$. Before presenting our transformation algorithm we make a couple of definitions.



a) Before the interchange b) After the interchange

Fig. 5.1: Type I interchange

Definition 5.1: A pair of nets, (a,b), in M_2^{TB} is said to form a *crossing pair* in an assignment if

i) Net "a" is connected by a path that does not cross the right side of G.
ii) Net "b" is connected by a path that does not cross the left side of G.
iii) On the bottom side of G all the terminals from net "b" appear to the left of all the terminals form net "a". And
iv) On the top side of G there is at least one terminal from net "a" located to the right of the rightmost terminal from net "b" and at least one terminal from net "a" is located to the left of the leftmost terminal from net "b".

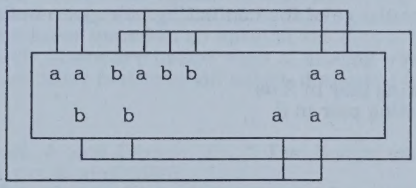


Fig. 5.2: Crossing pair (a,b).

One can easily prove that any two nets involved in a type I interchange will form a crossing pair. Later on we show that such a pair has harming effects only when it is a conflicting pair.

Definition 5.2: A crossing pair (a,b) *includes* point x if either of the two following conditions is satisfied:

i) If x is located on the top side of G, then all terminals from net "b" located on the top side of G appear to the left of x and the rightmost terminal from net "a" is located to the right of x. Or
ii) If x is located on the bottom side of G, then all terminals from net "b" located on the bottom side of G appear to the left of x and all the terminals from net "a" are located to the right of x.

Definition 5.3: A crossing pair (a,b) *partially includes* point x if x is located on the bottom side of G and either of the two following conditions is satisfied:

i) All terminals from net "a" located on the bottom side of G are located to the right of x and the leftmost terminal from net "b" located on the bottom side of G is not located to the right of point x. Or
ii) All terminals from net "b" located on the bottom side of G are located to the left of x and the rightmost terminal from net "a" located on the bottom side of G is not located to the left of point x.

Note that if point x is included in a crossing pair then it is also partially included in it, but the converse is not always true.

Definition 5.4: A *conflicting pair*, (a,b), is a crossing pair that includes the leftmost point with maximum height located on the top side of G, partially includes all the points with maximum height located on the bottom side of G and includes either the

leftmost or the rightmost point with maximum height located on the bottom side of G.

Our postprocessing procedure will find conflicting pairs and interchange their connecting paths. In figure 5.3 we show a conflicting pair and in figure 5.4 we indicate how these paths are interchanged. This transformation reverses the effects of type I interchanges.

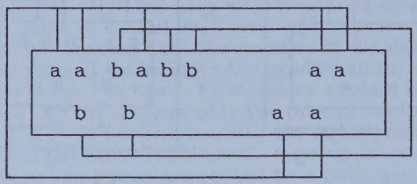


Fig. 5.3: (a,b) forms a conflicting pair.

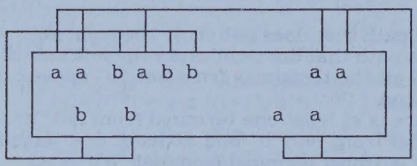


Fig. 5.4: Interchange of the conflicting pair given in figure 5.3.

algorithm MODIFY
  R ← S;
  while there is a conflicting pair in R *do*
    interchange a conflicting pair in R
  endwhile
end of algorithm

Let α be the number of type I interchanges made when transforming S to L and let $w + 3\alpha$ be the total increase in vertical height because of such transformation.

Lemma 5.2: The vertical height of assignment R minus the vertical height of assignment L is at most $w + 2\alpha$.

set	contribution to our lower bound for $h/(\lambda)+H_D(S_1,S_2)+H_D(S_3,S_0)$	contribution to our lower bound for $w/(\lambda)+H_D(S_0,S_1)+H_D(S_2,S_3)$
M_2^{TB}	m_2^{TB}	$2m_2^{TB}$
M_2^{LR}	$2m_2^{LR}$	m_2^{LR}

Table 5.1: Lower bounds for the R1M-2 problem.

An algorithm similar to MODIFY has to be applied to the $D_{i,j}$ in order to be able to claim the bound $1.6 * M_2^{LR}$. Let us refer to this new procedure MODIFY-LR. Our algorithm works as follows:

Obtain assignments $D_{i,j}$;
Use MODIFY and MODIFY-LR to transform each $D_{i,j}$ into a new assignment, $M_{i,j}$;
Output the assignment $M_{i,j}$ of least area;

Lemma 5.3: Let D be an optimal assignment such that $D' \subseteq D$. There is an assignment $M_{i,j}$ (constructed by our algorithm) such that if M is defined as D except for all the nets in $M_2^{TB} \cup M_2^{LR}$ which are routed as in the assignment $M_{i,j}$, then

a) $H_M(S_1,S_2) + H_M(S_3,S_0) \leq H_D(S_1,S_2) + H_D(S_3,S_0) + (0.6) * m_2^{TB}$

b) $H_M(S_0,S_1) + H_M(S_2,S_3) \leq H_D(S_0,S_1) + H_D(S_2,S_3) + (0.6) * m_2^{LR}$.

Lemma 5.4: Let D be an optimal assignment such that $D' \subseteq D$. Then assignment D and rectangle G satisfy the lower bounds given in table 5.1.

Theorem 5.1: Let D be an optimal assignment such that $D' \subseteq D$ and let P be the assignment produced by our algorithm. Then, $A(P) \leq 1.6 * A(D)$.

VI. APPROXIMATION ALGORITHM FOR THE R1M PROBLEM.

In this section we show that our algorithm takes $O(n^2)$ time and generates a solution with area $\leq 1.6 * OPT$, where OPT is the area of an optimal solution.

algorithm for the R1M Problem

Construct assignments D', D_3 and D_4;
$D_{i,j} \leftarrow D' \cup D_3 \cup D_4 \cup D_i^{TB} \cup D_j^{LR}$ for $0 \leq i,j \leq 63$;
Apply algorithm MODIFY and MODIFY-LR to each $D_{i,j}$ to obtain $M_{i,j}$;
Let P be one of the $M_{i,j}$'s of least area;
Construct and output a layout with area A(P) for P;

end of algorithm

Theorem 6.1: The time complexity of our procedure is $O(n^2)$.

Theorem 6.2: Let D be an optimal assignment such that $D' \subseteq D$ and let P be the assignment produced by our algorithm. Then, $A(P) \leq 1.6 * A(D)$.

Our algorithm generates 2^{12} assignments and it outputs one that requires the least layout area. An algorithm that only generates 2^6 assignments can be easily obtained by only taking the best of the modified $D_3 \cup D_4 \cup D_i^{TB}$ together with the best of the modified $D_3 \cup D_4 \cup D_i^{LR}$. For brevity we will not prove that this solution also satisfies our approximation bounds. In sections IV-V we assumed that the number of nets in some sets was a multiple of some fixed constant. More specifically, we might have to delete at most 5 nets from sets M_2^{TB} and M_2^{LR}; and at most 23 nets from M_4. All of these nets with a "small" number of terminals can be routed optimally by trying all possible routing paths and then selecting the best of the solutions generated. If some of these nets do not have a "small" number of terminals then select any routing paths for them, note that it will not make too much difference which routing path is selected since their contribution to the lower bound in an optimal solution is large (contribution from the number of terminals). There are better ways of dealing with the remaining 33 nets, however for brevity these other methods will not be discussed in this paper.

VII. REFERENCES

[AHU] Aho, A. , Hopcroft, J. and Ullman, J. , " *The Design and Analysis of Computer Algorithms* ", Addison-Wesley, 1975.

[GL1] Gonzalez, T. and Lee S., "An Optimal Algorithm for Optimal Routing Around a Rectangle," Proceeding of the 20th Annual Allerton Conference on Comm, Control and Computing, U. of Illinois, Oct. 1982, 636-645.

[GL2] Gonzalez, T. and Lee S., "An O(n log n) Algorithm for Optimal Routing Around a Rectangle," Technical Report #116, Programs in Computer Science, The University of Texas at Dallas, Nov 1982.

[GL3] Gonzalez, T. and Lee S., "Routing Multiterminal Nets Around a Rectangle," Technical Report #202, Programs in Computer Science, The University of Texas at Dallas, Nov 1983.

[GLL] Gupta, U. I., Lee D. T. and Leung J., "An Optimal Solution for the Channel-Assignment Problem", *IEEE Transactions on Computers*, Vol. C-28, No. 11, 1979, pp. 807-810.

[HS] Hashimoto, A. and Stevens J. E., "Wire Routing by Optimizing Channel Assignment without large Apertures", Proceedings of the 8th IEEE Design Automation Conference, 1971, pp 155-169.

[L] LaPaugh, A. S., "A Polynomial Time Algorithm for Optimal Routing Around a Rectangle", Proceedings of the 21st IEEE Foundations of Computer Science, 1980, pp 282-293.

[La] LaPaugh, A. S., "Algorithms for Integrated Circuit Layout, An Analytic Approach", Ph.D. dissertation, Massachusetts Institute of Technology, 1980.

[MC] Mead, C. and Conway, L., " *Introduction to VLSI Systems*", Addison- Wesley, 1980.

[R] Rivest, R.L., "The PI (Placement and Interconnect) System", Proceedings of the 19th IEEE Design Automation Conference, pp 475-481.

[SBR] Sahni, S., Bhatt, A. and Raghavan, R., "The Complexity of Design Automation Problems", Technical Report 80-23, Department of Computer Science, The University of Minnesota, Dec 1980.

AVERAGE CASE ANALYSIS OF MARKING ALGORITHMS — extended abstract

D.S. HIRSCHBERG* and L.L. LARMORE**
*University of California, Irvine, CA 92717
**California State University, Dominguez Hills, CA 90747

1. INTRODUCTION

Consider a set of nodes, each of which has two link fields that contain either the address of a node in the set, or an uninterpreted atomic value. One node is designated *root*.

In garbage collection, we may wish to mark all nodes accessible from the root; assume then that each node has a mark bit. Several algorithms to do this are known. Baer and Fries [1] analyze some of these, including the Schorr-Waite tag bit algorithm [3], the Wegbreit bit-stack algorithm [4] and the Lindstrom algorithm [2].

Both the Schorr-Waite and Wegbreit algorithms have time complexity $O(n)$ where n is the number of nodes accessible from the root, and both use $O(n)$ additional workspace. The Schorr-Waite algorithm requires a tag bit in each node whereas the Wegbreit algorithm uses a bit stack, whose size is $O(n)$ in the worst case, for the same purpose. It may appear that the maximum size of the Wegbreit bit stack is $O(\lg n)$ in the average case, but we show that it is $\Theta(n)$.

Lindstrom's algorithm uses bounded workspace, it uses no stack or tag bits. Its time complexity is $O(n^2)$ in all cases, but Lindstrom asserted [2] that the average case time complexity is $O(n \lg n)$. We prove that the average case time complexity is $\Omega(n^2)$.

2. THE TRACE TREE

The term *list structure* refers to the set of nodes accessible from *root*, together with their pointers. The various marking algorithms visit the nodes of a list structure in the same order (counting only first visits), called the *visitation order*. The *trace tree* τ of a list structure X is a binary tree with the same nodes as X but only a subset of the links. The father of node x in τ is the node in X from which x was visited in the visitation order. All other links of τ are λ.

$depth_\tau(P)$ is the depth of node P in τ. The time for the upward searching portion of the Lindstrom algorithm is $\Theta(\Sigma_{P \in X} depth_\tau(P))$. *hgt* is the height of τ.

Theorem 1. The time complexity of Lindstrom's algorithm is $\Omega(n \lg n + (hgt)^2)$.

3. AVERAGE CASE ANALYSIS

We assume a given distribution on the class of all list structures of size n. Let *probatom*(X) be the proportion of address fields in a list structure X which are atoms. Assume that the distribution satisfies the following condition.

Weak Randomness Assumption: There exist constants $0 \leq \alpha < 1/2$ and $\beta > 0$ such

This research was supported in part by National Science Foundation Grant MCS-82-00362.

that, if P is a new link and k is the number of nodes already visited:

(i) The probability P is an atom does not exceed α.

and (ii) The probability that P is an old node does not exceed $\beta k/n$.

Although we may allow knowledge of previously examined links to influence the expectation of P, we place a bound on that effect.

Theorem 2. In the average case, $hgt = \Theta(n)$.

Proof. We present an algorithm for visiting all nodes of X in visitation order (i.e., preorder for $\mathcal{T}$). Let S be a stack.

S $\leftarrow \emptyset$
P $\leftarrow$ *root*
While S is non-empty or P is a new node, do
    If P is an atom or an old node: P $\leftarrow$ top of stack S; pop S
    Otherwise: push RLINK(P) onto stack S; P $\leftarrow$ LLINK(P)

After t iterations of the loop, stack S will have some size, say $size(t)$, P will be some node, say $node(t)$, and P will be at depth $depth_{\mathcal{T}}$(P) in the trace tree. Each element on stack S is the right child of some node in the trace path of P. Therefore $depth_{\mathcal{T}}(node(t)) \geq size(t)$.

Think of *size* as a stochastic function. The number of old nodes after t iterations of the loop of the above algorithm is $\leq t$, and

$$size(t+1) = \begin{cases} size(t) + 1 \text{ if } node(t) \text{ is a new node} \\ size(t) - 1 \text{ otherwise} \end{cases}$$

By the Weak Randomness Assumption, the probability that $node(t)$ is a new node is at least $1 - \alpha - \beta t/n$, and the probability that $node(t)$ is not a new node is at most $\alpha + \beta t/n$. Let E(t) be the expected value of $size(t)$. Then

$$\begin{aligned} E(t) &= E(t\text{–}1) + \text{Prob}(node(t) \text{ is a new node}) - \text{Prob}(node(t) \text{ is not a new node}) \\ &\geq E(t\text{–}1) + (1 - \alpha - \beta t/n) - (\alpha + \beta t/n) \end{aligned}$$

Since $size(0) = 0$, we therefore have

$$E(t) \geq \Sigma_{1 \leq i \leq t} (1 - 2\alpha - 2\beta i/n) = t(1 - 2\alpha) - \beta t(t+1)/n$$

Now choose $t = \lfloor n(1\text{–}2\alpha)/(2\beta) \rfloor$. It is seen that the expected value of $size(t)$, and hence the expected value of $depth_{\mathcal{T}}(node(t))$, must be at least $n(1\text{–}2\alpha)^2/4\beta - 1$. □

Corollary. The size of the Wegbreit bit-stack is $\Theta(n)$ in the average case.

REFERENCES

[1] J.-L. Baer and M. Fries, *On the Efficiency of Some List Marking Algorithms*, Information Processing 77, IFIP, North-Holland Publishing Co., 1977, pp. 751-756.

[2] G. Lindstrom, *Copying List Structures in Bounded Workspace*, CACM 17 (1974), pp. 198-202.

[3] H. Schorr and W. M. Waite, *An Efficient Machine Independent Procedure for Garbage Collection in Various List Structures*, CACM 10 (1967), pp. 501-506.

[4] B. A. Wegbreit, *A Space Efficient List Structure Tracing Algorithm*, IEEETC, C-21 (1972), pp. 1009-1010.

THE CLOSEST NEIGHBOR PROBLEM (preliminary draft)*

WALTER CUNTO †
Departmento de Matimáticas y Computación, Universidad Simón Bolívar,
Caracas, Venezuela

and

J. IAN MUNRO ‡
Data Structuring Group, Department of Computer Science,
University of Waterloo,
Waterloo, Ontario, Canada, N2L 3G1

ABSTRACT

We show that $2n-3$ comparisons are necessary and sufficient in the worst case to find the largest element smaller than a specified element in a set of size n. $n+min(k,n-k)+o(n)$ comparisons are shown to be necessary and sufficient on average if the specified element turns out to be of rank k.

1. Introduction

An interesting but surprisingly little studied problem in selection is that of determining the nearest value in a set (presented as an unordered array) to a given value. The problem is quite easily solved in $2n$ or so comparisons in the worst case. But can this bound be improved for the worst case or perhaps on the average? In this paper we address just such problems.

More formally, we state the problems as: given a set X of n elements including a designated $x \in X$, then the *left neighbor* problem is that of finding the largest element in X that is less than x if such an element exists, and reporting the absence of such an element if it does not exist. The *right neighbor* problem is defined similarly, and the *both neighbors* problem is that of solving the elements above and below the given one, or finding one and reporting the absence of one of these if it does not exist.

A simple algorithm making at most $2n - 3$ comparisons will be shown to be optimal in the worst case. We also present an algorithm using $n + \min(k,n-k+1) + o(n)$ comparisons on average, where k denotes the (initially unknown) rank of x in X. Averaging over all possible ranks of x, an expected run time of $5n/4 + O(\log n)$ comparisons is obtained. These bounds are shown to be within the lower order term of optimal by adapting the approach used in [2] to prove

*This work was supported in part by the NSERC of Canada under grant A8237.
†The research discussed in this paper was done while the first author was at the University of Waterloo under the support of CEPET of Venezuela.
‡Currently on leave at AT&T Bell Laboratories, Murray Hill NJ.

lower bounds on the problem of finding the kth largest element.

2. Worst case optimality for closest neighbor(s)

An immediate algorithm for the problem is,

i) Partition X into $X_<$ and $X_>$, subsets of elements less and greater than x, respectively.
ii) Find the maximum of $X_<$ if the subset is not empty.
iii) Find the minimum of $X_>$ if the subset in not empty.

If $X_<$ and $X_>$ are both nonempty, the number of comparisons performed is $2n - 4$. Otherwise, if one of them is empty, only one neighbor is defined, and $2n - 3$ comparisons are performed. Note there are $n - 1$ elements in addition to x.

From the previous analysis we see:

Lemma 1: *Both neighbors of* $x \in X$, $| X | = n$, *can be found in* $2n - 3$ *comparisons in the worst case.*

This particular problem presents an interesting anomaly. If x turns out to have two neighbors, the algorithm performs faster (one comparison less) than in the case in which only one neighbor exists since x is an extremum of X. To take advantage of this anomaly, let us consider a simplified version of the problem, that of finding the left (smaller) neighbor of x, if it exists. Certainly the algorithm given above solves this variant, and any lower bound derived for the simplified version is a lower bound for the more general problem. By designing a simple adversary that resembles the one given in [1] for worst case selection, we will show that our algorithm is indeed optimal.

Theorem 2: *In the worst case,* $2n - 3$ *comparisons are necessary and sufficient to determine whether a left neighbor of* x *exists, and if so, to find it.*

Proof: The adversary will partition $X - x$, at any time, into five disjoint subsets.

i) X_v : the virgins, the elements having no comparisons associated with them,
ii) X_p : the pairs created from pairing virgins.
iii) $X_{2,>}$: the elements known to be greater than x, each element charged with two comparisons,
iv) $X_{2,<}$: the elements less than x, and charged with two comparisons per element.
v) $X_{1,<}$: the elements less than x, but larger than any element in $X_{2,<}$. Elements in this category are candidates for left neighbor, and each one of them has been charged with one comparison.

The only restriction on the adversary is that any given answer must be consistent. The relative order the adversary will always keep at any stage, is given in Figure 1. As an element is declared in $X_{2,<}$, it is declared to be smaller than any other element still in one of the other subsets. Similarly entry into $X_{2,>}$ declares an element larger than all remaining ones. $X_{2,<}$ and $X_{2,>}$ can be viewed as chains, which means that

even though the adversary provides the total order on those elements, no useful information is gained toward solving the problem at hand.

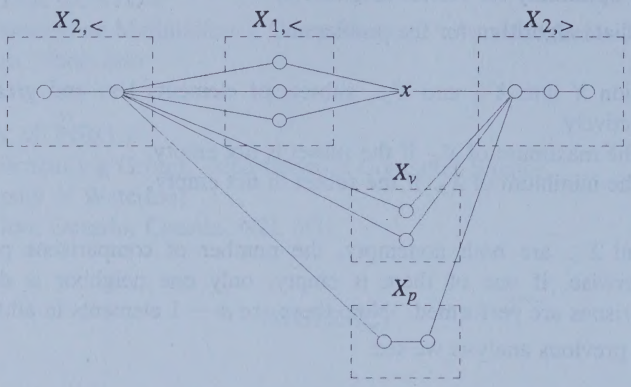


Figure 1
Partial order kept by the adversary

If the algorithm chooses to compare two elements a and b, the adversary will answer according to the following rules,

Case 1 : $a \in X_v$.

i) If a is compared with x, a is transferred to $X_{1,<}$.
ii) If b is another virgin, they are paired together and inserted into X_p.
iii) If b is the minimum of a pair in X_p with pair maximum denoted by c, b is declared to be in $X_{2,<}$ (although for consistency b is larger than any element previously put in $X_{2,<}$) and c becomes a 'reconstituted' virgin.
iv) If b is the maximum of a pair in X_p with pair minimum denoted by d, b is declared to be the minimum in $X_{2,>}$ and d becomes a virgin again.
v) If $b \in X_{1,<}$, then b becomes the maximum of those currently in $X_{2,<}$.

Case 2 : a is the minimum of a pair in X_p, with corresponding maximum denoted by c.

i) If compared with x, a is declared to be the maximum in $X_{2,<}$ and c becomes a 'reconstituted' virgin.
ii) If $b \in X_{1,<}$ or either element of a pair then, a becomes the maximum in $X_{2,<}$, and c a 'reconstituted' virgin.

Case 3 : a is the maximum of a pair in X_p with corresponding minimum denoted by c.

i) If compared with x, or either element of a pair, then a becomes the minimum in $X_{2,>}$ and c a 'reconstituted' virgin.
ii) If $b \in X_{1,<}$ then b becomes the maximum in $X_{2,<}$.

Case 4 : a, $b \in X_{1,<}$, b becomes the maximum in $X_{2,<}$.

Any other case leads to a redundant comparison. In particular $X_{2,<}$ elements always lose and $X_{2,>}$ always win.

The computation begins with all elements in X_v, and ends with all in $X_{2,<}$ or $X_{2,>}$ with the exception of one which may be in $X_{1,<}$. The latter condition occurs if x does have a left neighbor. The total charge to these $n - 1$ elements is, then, $2n - 3$.

3. Closest neighbor(s) average case

Now consider the average number of comparisons necessary and sufficient to solve the same problem.

Our algorithm is simple and keeps track of the closest neighbors found thus far on either side of x, together with a count on the number of values seen on each side. More formally:

i) Compare the first element with x, making it a neighbor candidate on the appropriate side.
ii) While all elements seen thus far have fallen on the same side of x, compare the next element with the current neighbor candidate and, if necessary, with x.
iii) After neighbors on both sides have been encountered, process each element by comparing it with the current neighbor on the more populous side of x.
iv) If necessary, compare it with the other neighbor.
v) If the new element falls between the current neighbors, compare it with x and replace the appropriate neighbor candidate with the new value.

This method uses at most $3n - O(1)$ comparisons; its average case behavior is more interesting and is sketched below.

Step i) is executed only once, the other $n - 2$ elements are first compared with a neighbor. The last part of step ii) and step v) are executed when a closer neighbor on either side has been encountered. This is expected to occur $H_{k-1} + H_{n-k}$ times (H_i denotes the sum of the reciprocals of the first i positive integers and is roughly $\ln i + \gamma$ where γ, Euler's constant, $\simeq .57721$). The interesting part of the analysis is how often step iv) is executed. Step iii) is essentially the process of drawing all $n - 1$ balls from an urn that initially contains $k - 1$ red and $n - k$ blue balls. We bet that the next ball seen will be the color of the majority seen thus far. If wrong, step iv) must be executed (step iv) is also executed if a new neighbor candidate is being processed, but that just doubles one of the logarithmic terms noted above). The crucial point turns out to be the number of times we have seen the same number of elements of each color and so don't know which way to bet. Consider two successive times at which the same number of red and blue balls have been drawn. If we bet correctly on the first draw in this period, then the same number of correct and incorrect bets would be made, and so it is irrelevant, for this period, which color is ultimately found to be more populous. Unfortunately, in such instances we can expect to be wrong half the time. Hence we must add one half the expected number of times the same number of balls of each color have been drawn. This is $O(n^{1/2})$ if $k = n/2 \pm o(n)$ but $O(1)$

otherwise. Thus we can conclude:

Theorem 3: *Let k denote the (initially unknown) rank of $x \in X$, $|X| = n$, then both neighbors of x can be found in $n + min(k,n-k) + O(log\ n)$ comparisons on average, unless $k = n/2 + o(n)$, in which case $n + min(k,n-k) + O(n^{1/2})$ suffice.*

We can adapt the approach used in the proof of the average case lower bound on finding the kth largest element [2] to the current problem. Again the lower bound will be shown for the easier problem of finding just the left neighbor, thus giving a stronger result. A sketch follows.

The key notions are those of close and straddle comparisons. A *straddle* comparison is one between elements on opposite sides of the desired value. Clearly $n - 1$ comparisons are necessary to verify a solution to the neighbor problem, and none of these can be straddles. Hence $n - 1$ plus any lower bound on the expected number of straddle comparisons is a lower bound on the problem at hand. In a verification, let alone an algorithm to find the desired element, every element must be compared with some element between it and the desired value. This fact, and the goal of showing that a some number of straddles are likely to occur, leads to the notion of a close comparison. The *close* comparison of the element of rank $k + i$ ($i \in [\ -min(k,n-k),\ min(k,n-k)],\ i \neq 0$) is the first between it and that of rank $k + j$ ($j \in [-i,\ i-1]$ or $[i+1,\ -i]$), i.e. the first between an element and some value at least as close in rank to the desired one.

Lemma 4 (adapted from [2]): *For any algorithm to solve the left neighbor problem, the expected number of close comparisons that are straddles for the element of rank $r + 1 + i$, if the desired element is of rank $r + 1$, plus those for the element of rank $r - i$, if the desired element is of rank r, is 1.*

It is possible that a comparison may be the close comparison of both elements involved, call such comparisons *doubleclose*. As noted above, each element must be compared with the desired element or some value between these two. Call the first such comparison involving a given element its *crucial* comparison.

Lemma 5 (adapted from [2]): *For any algorithm to solve the left neighbor problem, the expected number of comparisons that are neither close nor crucial is at least the expected number of double close comparisons.*

We are now ready to state our lower bound.

Theorem 6: *The average number of comparisons necessary to find the left neighbor of a given value in a set of size n is at least $n + min(k,n - k) - O(1)$, where k denotes the rank of the given element in the set.*

Sketch of Proof: The theorem follows from the lemmata by including the dummy element $-\infty$ for the rank $k - 1$ case and $+\infty$ for the k case.

Corollary 7: *$5n/4 + O(n^{1/2})$ comparisons are necessary and sufficient for computing the closest neighbors of a randomly picked element from a set X of cardinality n.*

Proof: Follows from Theorems 3 and 6 by averaging over all possible ranks of x.

1. Conclusions

We have shown rather tight bounds on the number of comparisons necessary to determine the neighbors of a given value. Clearly the upper bound on the average case can be reduced by one of the logarithmic terms if only one neighbor is required. The most obvious questions left open are those dealing with lower order terms. Can the $O(n^{1/2})$ term be removed when $k \simeq n/2$? Can the H_n terms be reduced to $O(\text{loglog } n)$, as is the case in determining the second largest element of a set([5],[6])? Answers to either of these questions would be interesting. Unfortunately, positive answers would probably come from impractical algorithms. One of the nice features of our study, thus far, has been that the algorithms have been clean and easily coded.

2. References

[1] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest and R.E. Tarjan, "Time Bounds for Selection", *J. Computer and Systems Sciences* 7:4 (1973), 448-461.

[2] W. Cunto and J.I. Munro, "Average Case Selection", *Proc 16th Annual ACM Symp. on Theory of Comp.* (1984), 369-375.

[3] W. Feller, *An Introduction to Probability Theory and its Applications, Volume I*, Wiley, 1950.

[4] R.W. Floyd and R.V. Rivest, "Expected Time Bounds for Selection", *CACM* 18 (1975), 165-172.

[5] D. Matula, *Selecting the t-th Best in Average n + O(lglg n) Comparisons*, Washington University (St. Louis) Mathematics Dept. TR 73-9 (1973).

[6] A.C. Yao and F.F. Yao, "On the Average-case Complexity of Selecting the k-th Best", *SIAM J. on Computing*, 11 (1982), 428-447.

A FULLY-POLYNOMIAL APPROXIMATION SCHEME FOR VARIABLE SIZED BIN PACKING

FRANK D. MURGOLO
University of California, Irvine, CA 92717.

0. ABSTRACT

In the typical bin packing problem one assumes that all bins are of the same size. In a more general setting one is given a set of different bin sizes and wishes to pack a list of n items into a set of bins which minimizes the sum of the sizes of the bins used. We show that for every positive ϵ there exists an algorithm A, linear in n, such that if $A(L)$ represents the sum of the bin sizes used by A to pack a list of items L then $A(L) \leq (1+\epsilon)\ OPT(L)+O(1/\epsilon^4)$. We then provide an algorithm which runs in time polynomial in both n and ϵ.

1. INTRODUCTION

In the standard bin packing problem, one is given a list of items a_1, a_2, a_3 $\cdots$, a_n and asked to pack them into the minimum number of bins so that the total size of the items in any single bin does not exceed 1. Because this problem is *NP–Complete*, researchers have spent considerable effort in devising time efficient approximation algorithms. For a given approximation algorithm, A, and an instance I of the bin packing problem we define r_I, the performance ratio of A, as the ratio $\frac{A(I)}{OPT(I)}$. The asymptotic performance ratio, r, is the least positive r such that for some positive integer N, $r_I \leq r$ for all instances I satisfying $OPT(I) \geq N$. A polynomial approximation scheme for a bin packing problem takes as input an instance I of the problem and an error bound ϵ and yields, for each fixed ϵ, a polynomial time solution with the property $r \leq 1+\epsilon$. A fully polynomial approximation scheme is one which is also polynomial in $1/\epsilon$. Until recently, the best value for r was 1.183... [GJ]. [DL81] provides a polynomial approximation scheme with a ratio of $1+\epsilon$ for any $\epsilon > 0$ which was extended in [KK82] to a fully polynomial approximation scheme.

In the variable size bin packing problem, one is given a set of bins of various sizes and a list L of items that are to be packed into bins whose sizes are drawn from the given set. The objective is to choose the bins in such a way as to minimize the sum of the sizes of the bins used. An instance I of this problem can be considered as a list L of items to be packed along with the set of bin sizes available for the packing. For an algorithm A that provides a packing for I, we let $A(I)$ denote the sum of the bin sizes used by A to pack L. We use $OPT(I)$ to denote the minimum sum of bin sizes attainable for I.

Supported by National Science Foundation Grant MCS 79-04997

An early industrial application of this problem [Ei57] is the cutting stock problem in which one has a set of different sized pieces of stock from which an order for various lengths must be cut. [GG61] examines a linear programming solution to this problem which derives approximate solutions efficiently in practice. The variable size bin packing problem is *NP–Complete*, thus leading one to search for efficient approximation schemes. [FL83] describes approximation schemes which provide performance ratios of 2, 3/2 and 4/3, respectively. In this paper we prove that given a problem instance I and $\epsilon > 0$ there exists a linear time algorithm A such that $A(I) \leq (1+\epsilon)OPT(I)+O(1/\epsilon^4)$, which provides an asymptotic performance ratio of $1+\epsilon$. We employ methods similar to those in [DL81] and [KK82], with different techniques for handling small items and new techniques for dealing with variable sized bins. In order to attain an algorithm with time complexity polynomial in ϵ we describe how we can make use of the *Fractional Bin Packing Algorithm* of [KK82] to solve the linear programming formulation of our problem.

This paper is organized as follows: Section 2 describes the algorithm, Section 3 describes the amount of error caused by the rounding of items sizes, Section 4 describes the efficacy of the linear programming formulation as well as the error introduced when padding out the linear solution to integers, and Section 5 bounds the amount of error introduced by disregarding small items and small bins. Finally, Section 6 describes the time complexity of the algorithm.

2. THE APPROXIMATION SCHEME

The approximation scheme is comprised of the following steps:

1. Defining a minimum item size and bin size which are both functions of ϵ.
2. Rounding the items sizes in order to decrease the number of different item sizes to be packed.
3. Temporarily disregarding small items and small bins and formulating the packing problem as a linear programming problem.
4. Taking the packing provided by the linear program solution and creating a proper packing of all the items, i.e. padding out the solution to integers and packing the small items.

We assume we are given an instance I of the variable size bin packing problem made up of:

1) *BINS*: a set of available bin sizes, (as in [FL83] we assume, wlog, the largest bin size is 1),
2) L: a list of items (and sizes) $a_1, a_2, \cdots, a_n$,
3) $\epsilon > 0$ where we desire a performance ratio of $1 + \epsilon$.

The performance measure of a variable size bin packing algorithm A for an instance I can be formally defined as:

$$A(L,BINS,\epsilon) = \sum_{i=1}^{B} s(B_i)$$

where:

B = the number of bins used by A,

B_i = the i^{th} bin used,

$s(B_i)$ = the size of B_i.

As a notational convenience, we will write $A(L,BINS,\epsilon)$ as $A(L)$. (We note this is the same as the performance measure for the bin packing problem where all bins have a size of 1 and one counts the number of bins used as the performance measure.)

We begin by defining: i) $min = \epsilon$; ii) $\delta = \epsilon \times min = \epsilon^2$. We call items of size less than δ 'small items' and bins of size less than min 'small bins'. These small items and small bins are temporarily excluded from consideration. We will eventually pack the small items but we never use the small bins.

3. ROUNDING OF ITEM SIZES

As we will see later, the number of different item sizes plays an critical role in determining the time complexity of the approximation scheme. For this reason , using methods similar to those in [DL81] & [KK82], we limit the number of distinct sizes by a rounding procedure and show that this rounding does not affect the performance ratio adversely.

At this stage of the approximation scheme the list to be packed contains no small items and $BINS$ contains no small bins. Let L_0 be this new L and let m be the number of items in L_0. We assume the list $L_0 = a_1, a_2, \cdots, a_m$ has the property: $s(a_1) \geq s(a_2) \geq \cdots \geq s(a_m)$. For some k, which we determine later, we round every k items up in size to the size of item a_{kj} for $j = 0,1,2,\ldots,\lfloor m/k \rfloor$, setting $a_0 = 1$. For example, the sizes of the first k items $a_0, \cdots, a_{k-1}$ are rounded to the sizes of $a_{k0} = a_0 = 1$. The size of items a_k thru a_{2k-1} are rounded to the size of a_k. Calling this modified list L' we ask, 'How does a packing of L' differ from a packing of L_0 ?' Using a packing of L' one can easily construct a packing for L_0 using no more bins than used for L' yielding:

$$OPT(L_0) \leq OPT(L')$$

Next we imagine, for purposes of analysis, rounding every successive k items down in size to a_{kj} for $j = 1,2,3,\ldots,\lceil m/k \rceil$. (If m/k is non-integer we can pad out the list with items of size 0). Calling this set $\hat{L}$ we have

$$OPT(\hat{L}) \leq OPT(L_0) \tag{1}$$

Now we observe that L' is identical to $\hat{L}$ except L' has k items of size 1 and $\hat{L}$ may have some additional items of size $a_{\lfloor m/k \rfloor k}$. This tells us

$$OPT(L') \leq OPT(\hat{L}) + k \tag{2}$$

Together (1) and (2) imply:

$$OPT(L') \leq OPT(L_0) + k$$

which implies

$$\frac{OPT(L')}{OPT(L_0)} \leq 1 + \frac{k}{OPT(L_0)}$$

Since every item being packed has size greater than δ we have $OPT(L_0) \geq \delta m$, which together with the above yields

$$\frac{OPT(L')}{OPT(L_0)} \leq 1 + \frac{k}{\delta m}$$

We therefore set $k = \lceil \delta^2 m \rceil$ and derive:

LEMMA 1: $\frac{OPT(L')}{OPT(L_0)} \leq 1 + \delta \leq 1 + \epsilon$.

Hence we see that by packing L' instead of L_0, the relative error introduced is at most ϵ.

4. AN ALGORITHM TO PACK L'

Given the list of L', as described in the previous section, we have only $S = \lfloor m/k \rfloor + 1$ different item sizes with at most k items of each size.

Let $OPT_\epsilon(items)$ be an optimal packing of the set *items* using only bins of size greater than ϵ. Define a *configuration* to be a packing of a specified bin size with a specified set of item sizes such that the sum of the item sizes does not exceed the size of the bin. Because we are packing only items of size greater than δ, we know at most $\lfloor 1/\delta \rfloor$ items can fit into a bin thereby limiting the number of possible configurations to a number N which is independent of n. We consider the list L' as the multiset

$$L' = \{n_1 a_1, n_2 a_2, \cdots n_S a_S\}$$

where $n_i a_i$ means item a_i is of multiplicity n_i.

Define:

1) f_{ij} to be the number of items of size $s(a_i)$ that are used in the j^{th} configuration.
2) c_j as the size of the bin used in configuration j, also called the *cost*.
3) x_j as the number of times configuration j is used in a given packing.

We have the following integer program:

$$\textit{Minimize:} \quad \sum_{j=1}^{N} c_j x_j$$

$$\textit{Subject to:} \quad f_{i1}x_1 + f_{i2}x_2 + \cdots + f_{iN}x_N \geq n_i, \qquad \text{for } 1 \leq i \leq S.$$

$$\text{and} \quad x_j \geq 0, \qquad 1 \leq j \leq N.$$

We relax the restrictions to integers and solve the above as a general linear programming problem only constraining the x_i to be non-negative reals. Call the solution to this problem $LIN(L')$. Relaxing the constraints can only lower the

minimum so $LIN(L') \leq OPT_\epsilon(L')$. It is a fact of linear programming that for a problem with N variables and S constraints there exists a solution with at most S non-zero values for the variables. Therefore, we can obtain a solution for which each x_i is an integer value by adding no more than S bins. Call this $PAD(L')$. Therefore, $PAD(L') = LIN(L') + S \leq OPT_\epsilon(L') + S$. Hence we have the following:

LEMMA 2: $PAD(L') \leq OPT_\epsilon(L') + O(1/\epsilon^4)$.

5. PACKING SMALL ITEMS WHILE EXCLUDING SMALL BINS

We now describe how our algorithm, *ALG*, provided by our scheme for the instance I deals with the small items. Taking the packing derived by the previous steps use a simple First-Fit strategy and pack as many small items as possible into the already allocated bins. If there are any small items that can not fit into the already allocated bins, pack them into new unit size bins, again using a First-Fit strategy.

We will show that this simple packing of the small items does not harm the approximation derived by *PAD*. In particular this shows one can exclude the small bins from the packing and still achieve the desired approximation. We refer to the algorithm provided by our scheme for an instance I as $ALG(L)$. L denotes the original list of items.

THEOREM 1: For a given instance I, $ALG(L) \leq (1+\epsilon+2\epsilon^2)OPT(L)+O(1/\epsilon^4)$
proof

We have two cases to consider:

[1] **No new bins are allocated when packing small items.**

In order to analyze this case, we introduce another packing which is not part of our algorithm but is helpful in deriving the relationship between $OPT(L)$ and $PAD(L)$.

Given an optimal packing of L using all bin sizes of I, we create a new packing $ALG'(L)$ using only bins of size greater than ϵ by taking all the items which were packed by $OPT(L)$ into small bins and repacking them into unit size bins using a First-Fit strategy. Before we describe how $ALG'(L)$ relates to $OPT(L)$ and $PAD(L)$ we first give some definitions.

1. $OPT(L)$ = optimal packing of the list L using all bin sizes.
2. B_l = set of bins of size greater than ϵ used in $OPT(L)$.
3. B_s = set of bins of size less than or equal to ϵ used in $OPT(L)$.
4. $L_l = L$ restricted to those items packed into B_l in $OPT(L)$
5. $L_s = L - L_l$.
6. $s(L_l) = \sum_{a_i \epsilon L_l} s(a_i)$; with $s(L_s)$ defined similarly.
7. $s(B_l) = \sum_{B_i \epsilon B_l} s(B_i)$; with $s(B_s)$ defined similarly.

Letting $k+1$ be the number of new unit size bins created by $ALG'(L)$ when repacking the small items, we can write

$$ALG'(L) \leq s(B_l) + k + 1 \tag{1}$$

We observe that at least k of the new bins are filled to within ϵ of their capacity which allows us to bound k,

$$S(L_s) \geq (1-\epsilon)k$$

so,

$$\frac{s(L_s)}{1-\epsilon} \geq k$$

which together with (1) yields:

$$ALG'(L) \leq s(B_l) + \frac{s(L_s)}{1-\epsilon} + 1 \tag{2}$$

It is certainly true that $s(L_s) \leq s(B_s)$ which taken with (2) yields:

$$\begin{aligned} ALG'(L) &\leq s(B_l) + \frac{s(B_s)}{1-\epsilon} + 1 \\ &\leq \frac{s(B_l)}{1-\epsilon} + \frac{s(B_s)}{1-\epsilon} + 1, \quad \text{for } \epsilon < 1. \\ &\leq \frac{1}{1-\epsilon}\,[s(B_l) + s(B_s)] + 1 \end{aligned} \tag{3}$$

Now:

$$OPT(L) = s(\textit{bins used by } OPT) = s(B_l) + s(B_s) \tag{4}$$

Equations (3) & (4) imply:

$$ALG'(L) \leq \frac{OPT(L)}{1-\epsilon} + 1$$

or

$$ALG'(L) \leq (1+\epsilon+2\epsilon^2)OPT(L) + 1 \tag{*}$$

for sufficiently small ϵ.

We now show how $ALG'(L)$ compares to $PAD(L)$. By Lemma 2 and the assumption of this case we have

$$PAD(L) \leq OPT_\epsilon(L) + O(1/\epsilon^4). \tag{5}$$

Because $ALG'(L)$ is an algorithm for packing L into bins of size greater than ϵ we know $OPT_\epsilon(L) \leq ALG'(L)$ which together with (5) tells us:

$$PAD(L) \leq ALG'(L) + O(1/\epsilon^4) \tag{6}$$

By the assumption of this case, $ALG(L) = PAD(L)$, which taken with (6) and (*) gives us the desired result:

$$ALG(L) \leq (1+\epsilon+2\epsilon^2)OPT(L) + O(1/\epsilon^4)$$

[2] **New bins are allocated when packing small items.**

In this case we know:

$$\sum_{i=1}^{n} s(a_i) \geq (ALG(L) - s(last\ bin)) - \delta(B-1), \qquad (1)$$

$$\geq (ALG(L) - s(last\ bin)) - \delta\, B$$

where B is the number of bins used by ALG. Clearly, $B \leq \frac{ALG(L)}{\min}$ which then gives

$$\sum_{i=1}^{n} s(a_i) \geq (ALG(L) - s(last\ bin)) - \delta\frac{ALG(L)}{\min}$$

$$\geq (1-\frac{\delta}{\min})ALG(L) - s(last\ bin)$$

$$\geq (1-\epsilon)ALG(L) - s(last\ bin)$$

by the definition of δ. Certainly, $OPT(L) \geq \sum s(a_i)$, so we derive:

$$ALG(L) \leq \frac{OPT(L)}{1-\epsilon} + \frac{s(last\ bin)}{1-\epsilon}$$

or

$$ALG(L) \leq (1+\epsilon+2\epsilon^2)\ OPT(L)+O(1/\epsilon^4) \quad \text{for } \textit{sufficiently small } \epsilon.$$

This completes our proof.

Hence, by Lemma 1 and Theorem 1 we know that for any instance I of a variable bin packing problem and any $\epsilon > 0$ there exists an approximation algorithm ALG with the property

$$ALG(L) \leq (1 + \epsilon + 2\epsilon^2)OPT(L) + O(1/\epsilon^4)$$

Note that the factors of $2\epsilon^2$ and $1/\epsilon^4$ do not affect our result but are present for notational ease. We can choose an ϵ' as a proper function of ϵ to achieve the stated result since the bound is to hold for arbitrarily small ϵ.

6. TIME COMPLEXITY

(Recall n is the number of items in a list L to be packed.) Defining the minimum bin and item size certainly takes a constant amount of time. To define the item sizes for the list L' we can sort the list for a time of $O(nlogn)$ or as suggested in [DL81] we can use the solution for the 'selection problem' and find the special sizes $a_0, a_k, a_{2k}, \ldots, a_{\lfloor n/k \rfloor k}$ in time $O(n)$ with a constant that grows quickly with ϵ. The solution to the linear program is achieved in time independent of n while the first-fit packing of the small items is done in linear time. Therefore we have a linear time polynomial approximation scheme.

To extend this scheme to one which is polynomial in $1/\epsilon$ we make the following observations. To round the items we first sort them taking time $O(nlogn)$. The time to solve the linear program is a function of S and N, which are independent of n but functions of ϵ. In order to achieve a running time

which is polynomial in $1/\epsilon$ we make use of the *Fractional Bin-Packing Algorithm* described in Section 6 of [KK82]. The linear program we have formulated herein is identical to the formulation found in [KK82] with the following difference. In [KK82] their objective function has a unit cost matrix whereas ours is comprised of the variable bin sizes used in the set of configurations.

Let $C = (c_1, c_2, \cdots, c_N)$ where c_i is the cost of the bin used in the i^{th} configuration. In examining their algorithm we make the following modifications.

Their object is to solve within an additive error the linear program

$$\textit{minimize} \quad 1x \quad \textit{subject} \quad x \geq 0, \quad Ax \geq b.$$

where the columns of A correspond to our f_{ij}'s which are derived from the possible bin configurations. In our formulation we ask to

$$\textit{minimize} \quad Cx \quad \textit{subject} \quad x \geq 0, \quad Ax \geq b.$$

Because this linear program has a large number of variables one considers the dual linear program which has a constraint for each bin configuration but only S different variables. We can solve the dual and use the solution for the original problem due to the duality theorem of linear programming which says the primal and dual programs have the same optimum. In [KK82] they have the dual program : maximize ub subject to $u \geq 0$ and $u^T A \leq 1$. The dual program has the following economic interpretation. The dual variable u_i is the price of each piece of type a_i. The program seeks to maximize the sum of the prices of all the items, subject to the constraints that each price be nonnegative and the sum the prices of all the pieces in any single bin be less than or equal to 1.

In our case, we have the dual program : maximize ub subject to $u \geq 0$ and $u^T A \leq C$. We must insure that the sum of the prices of the items in any configuration are less than the cost of the bin used in that configuration. (Recall that the cost of a bin is its size.)

In order to test the feasibility of a price vector u the algorithm in [KK82] uses an *approximate feasibility* test based on the solution to a particular knapsack problem. At each iteration an ellipsoid with center u is given. The vector u is rounded to a nearby vector u' where u' is feasible if and only if the optimal value of the following knapsack problem is ≤ 1: maximize vu subject to $vs \leq 1$, $v \geq 0$, v an integer. Here v is a vector representing a bin configuration containing v_i items of type i, and s is a vector whose i^{th} component is the size of an item of type i. (The vector u' is used instead of u so the knapsack problem is easy to solve. This approximation does not harm the desired error bound.) In our case we ask to maximize vu subject to $vs \leq C$ i.e., we need to have the sum of the sizes of the items used in the i^{th} configuration be less than or equal to the size of the bin used in the i^{th} configuration. The solution by dynamic programming is then checked to see if its value is greater than the bin size, c_i, of the configuration being solved for.

The primary reason we are able to use the *Fractional Bin Packing Subroutine* of [KK82] is that the dual of our problem has constraints of the form

$$f_{1i}u_1 + f_{2i}u_2 + \cdots + f_{Si}u_S \leq \beta_i$$

where β_i is the size of the bin used in the i^{th} configuration. In [KK82] the β_i are all 1 whereas in our application we can have different values for the β_i. The fortunate feature is that the formulation of the problem has allowed for inclusion of the size of the bin used in a particular configuration to appear explicitly in the dual linear program.

When analyzing the time complexity of our use of this modified ellipsoid method we need to consider the affect variable size bins have on *constraint elimination*. Because we are solving the dual of our original linear program the number of constraints is the number of different bin configurations and the number of variables is the number of distinct item sizes. We can possibly have more bin configurations due to the variable bin sizes but the increase will only be polynomial thereby allowing us to retain the desired time bound.

Using these observations along with facts taken from [GLS81] one can show that the linear programming formulation of our problem can be solved in time polynomial in $1/\epsilon$. Therefore, we can achieve an approximation scheme for variable sized bin packing which is polynomial in $1/\epsilon$ as well as n.

7. ACKNOWLEDGEMENTS

I wish to thank George Lueker for many helpful comments and suggestions. I also would like to thank Scott Huddleston and Dennis Volper for pointing out areas of possible confusion in an early draft.

REFERENCES

[DL81] Fernandez de la Vega, W. and Lueker, G.S., *Bin packing can be solved within 1 + ϵ in linear time,* Combinatorica 1 (1981), 349-355.

[Ei57] Eisemann, K. *The Trim Problem,* Management Science, 3, 1957, 279-281.

[FL83] Friesen, D.K. and Langston, M.A., *Variable Sized Bin Packing,* Technical Report CS-83-114, July 1983, Washington State University; Pullman, Washington 99164.

[GG61] Gilmore, P.C. and Gomory, R.E., *A linear programming approach to the cutting stock program,* Operations Research, 9 (1961), 849-859.

[GJ] Garey, M.R. and Johnson, D.S., paper in preparation.

[GLS81] Grotschel, M., Lovasz, L. and Schrijver, A., *The ellipsoid method and its consequences in combinatorial optimization,* Combinatorica 1 (1981) 169-198.

[KK82] Karmarkar, N. and Karp, R.M., *An Efficient Approximation Scheme For The One-Dimensional Bin-Packing Problem.,* Proceedings 23rd Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1982, 312-320.

The Design and Analysis of BucketSort

(extended abstract)

Eugene E. Lindstrom
IBM Palo Alto Scientific Center
1530 Page Mill Road
Palo Alto, CA 94304

Jeffrey Scott Vitter
Department of Computer Science
Brown University
Providence, RI 02912

ABSTRACT

BucketSort is a new external sorting algorithm for very large files that is a substantial improvement over merge sorting with disks. BucketSort requires an associative secondary storage device, which can be realized by large disk drives with logic-per-track capabilities or by magnetic bubble memory (MBM). This paper describes and analyzes a hypothetical BucketSort implementation that uses bubble memory. A new software marking technique is introduced that reduces the effective time for an associative search.

One of the important features of BucketSort is that the algorithm is randomized; there are no worst-case input files. For purposes of analysis, no assumptions are made about the distribution of the key values or about the initial order of the keys. Two examples are studied for purposes of comparing BucketSort with conventional merge-based methods. The sorting time for the two examples using an optimized merge sort with disk storage is about one hour and 10-15 hours, respectively; BucketSort needs only about 5-10 minutes and 1-2 hours. The sorting time can be cut substantially by using more internal memory. It can be reduced further by using faster I/O channels if the sort is I/O-bounded, or by using a vector processor if the sort is CPU-bounded. BucketSort can be a natural addition to relational database systems.

1. INTRODUCTION

The BucketSort algorithm, which was introduced in [Lindstrom and Vitter 81] and [Lindstrom and Vitter 83], is a new external sorting method for very large files that significantly outperforms the current methods. BucketSort requires an *associative* secondary storage device, which can be approximated by large disk drives with logic-per-track capabilities or by magnetic bubble memory (MBM). A disk implementation was considered in [Lindstrom and Vitter 81], and subsequent modifications appear in [Johnson and Nishimura 84]. This paper gives the design and analysis of a hypothetical BucketSort implementation that uses MBM. The complete details are given in the full paper [Lindstrom and Vitter 84].

The BucketSort algorithm is *randomized*: it takes random actions during the course of execution, and hence its running time for any given input file is a random variable. An important feature of BucketSort is that the distribution of the running time is *independent* of the initial order of the keys and of the distribution of the key values. There are no worst-case input files. A sorting method for associative secondary storage was proposed in [Lin 77], but it works well only when the key values are independent and identically distributed, which is not a practical assumption. The success of BucketSort is due to the randomization in the algorithm and to the marking technique described in Section 5, as well as to the use of associative secondary storage.

A complete description of the algorithm appears in the next section. Section 3 explains how to set two important parameters in BucketSort: the size of the sample taken during the Sample Phase and the minimum internal memory size. The analysis is omitted for lack of space. In Section 4, we discuss how to implement associative secondary storage by use of magnetic bubble memory. The time per associative search is on the order of 1-2 seconds. Section 5 describes a new software marking technique that can reduce the time for each bucket retrieval during the Internal Sort Phase to only a fraction of a second.

In Section 6, we list the file sizes and internal memory requirements for two very large sorting examples. The first example is typical of the large sorts done once each night by major banks; the second represents the projected file sizes for the large sorts expected within this decade. We calculate the execution times of BucketSort for the two examples and compare them with a highly optimized merge sort. BucketSort is approximately 10 times faster.

Since this sorting method is partition-based, the external nature of the BucketSort algorithm is almost independent of the internal sorting of each bucket. Section 7 discusses ways to speed up the Internal Sort Phase (when the CPU turns out to be the bottleneck) by using more internal memory and a vector (or array) processor. We show that if certain vector instructions are available, internal sorts can be done about four times faster. Section 8 discusses the use of associative secondary storage MBM for storing relational databases. BucketSort can be viewed as one of the operations in such a database. In Section 9, we summarize our results and draw conclusions.

2. THE BUCKETSORT ALGORITHM

BucketSort is an example of a distribution sorting algorithm. In the full version of the paper, we will show why distribution sorting methods are inferior to merge sort when conventional disk storage is used. The success of BucketSort is due to the use of associative secondary storage and to the randomization in the algorithm. There are no worst-case input files; all the randomization is contained within the structure of the algorithm itself. The sorting method operates in the three phases described below. Modifications will be discussed in the full paper. We use the following notation:

N = number of records
M = internal memory size (bytes)
R = record size (bytes)
K = key size (bytes)
S = number of sampled keys $\cong M/K$
F = number of records that can be stored internally $\cong M/R$

For convenience, we assume that the records are fixed-length. Modifications to handle variable-length records will be covered in the full paper.

Sample Phase

A random sample of $S \cong M/K$ keys is taken, using one of the sampling methods discussed in [Knuth 81] and [Vitter 84]. The size of the sample is an important parameter that is discussed in the next section. The sample is sorted internally using a balanced search tree, such as an AVL-tree or an RB-tree, so that the sorting time overlaps with the sampling as much as possible. Each node in the balanced tree requires storage space for the key, plus two pointers for the sorting.

The magnetic bubble memory (MBM) is made up of small units called arrays, as discussed in Section 4. In the case of sequential files that must pass through the CPU in order to get stored on the associative MBM, each record is stored in a random array in the MBM. The sampling is done while the records are being distributed on the MBM. We can force a stable sort by appending a sequence number to the end of the key field. The actual key field might be scattered in parts throughout the record, each part in a different data format. For purposes of speeding up the sort, it may be desirable to convert each key field to binary notation and to reformat the record so that the binary key value appears at the front of the record.

If the records of the input file already reside in the MBM, then we assume that they were placed on random arrays; if not, the records should be randomly scrambled among the arrays so that the marking technique described in Section 5 can be used most effectively. We can do the sampling directly from the bubble memory at the expense of more logic complexity. First, we randomly decide how many of the keys will be picked from each array of the MBM, based on the multinominal distribution. Then we use one of the sampling methods in [Knuth 81] or [Vitter 84] in order to sample from each array in parallel.

The last part of this phase consists of an inorder traversal of the balanced tree in order to create a sequential list of pointers that point to the keys in the tree in increasing order.

Bucket Formation Phase

This phase is divided into two parts: the counting subphase and the combination subphase. The S sampled key values that were sorted in the Sample Phase break up the file into $S + 1$ *minibuckets*. Let us denote the sorted key values by

$$X_1, X_2, \ldots, X_S;$$

we also define $X_0 = -\infty$ and $X_{S+1} = +\infty$. For each value of i, $1 \geq i \geq S + 1$, the i^{th} minibucket is defined to be the set of records whose key value is in the range

$$X_{i-1} < (\text{key value}) \leq X_i.$$

Each minibucket contains an average of about N/S records with a standard deviation of about N/S. In the next section, we discuss how to choose S so that N/S is very much smaller than F, which is the the number of records that can fit in internal memory.

The sorted order of the keys is represented by the sequential pointer list that was created at the end of the Sample Phase. The keys are still stored in the balanced tree. The counting subphase consists of processing each key on the associative device and performing a binary search on the pointer list in order to determine which small bucket the key belongs to. The count for that bucket is incremented by one.

In the combination subphase, contiguous minibuckets are grouped together to form larger *buckets*. Each grouping combines as many minibuckets as possible so that the resulting bucket contains at most F records; the number of records in each bucket is in most cases approximately equal to F, as shown in Section 3. Each minibucket that already contains more than F records is called an overflow bucket; it is proved in Section 3 that the average number and the standard deviation of the number of overflow buckets is less than 1.

The keys that define the ranges for the buckets are output sequentially to a conventional disk or tape (unless there is room in the internal memory). In the case of an overflow bucket, the count (or else a special marker) is output along with the key that defines the upper limit of the bucket.

Internal Sort Phase

In this phase, the keys that define the bucket ranges are processed sequentially. For each bucket range, the records in that bucket are retrieved by an associative search, sorted, and appended to the output file. The mechanism by which each bucket is retrieved is described in Section 5 and in the full paper. Study shows that the channels remain 98-99% busy; the overhead is therefore negligible.

If the bucket is not an overflow bucket, all its records fit in internal memory and can be sorted internally. The Replacement Selection method is a natural choice for sorting the buckets, because there is no need to reinitialize for each bucket: when one bucket has been retrieved from secondary storage, retrieval of the next bucket can begin, even though some of the former's records may still be in the Replacement Selection tree. This works because all the key values from the first bucket are less than or equal to all the key values from the second bucket, and because each bucket's records can fit in the tree at one time.

If the bucket is an overflow bucket, then it can be sorted either by applying BucketSort recursively or by using a conventional merge sort. The extra time needed to handle overflows is insignificant, since the number of overflow buckets is almost always roughly 1.

3. CHOICE OF SAMPLE SIZE S AND MEMORY SIZE M

One of the key ideas in BucketSort is the way in which the number S of sampled keys and the required amount M of internal memory are determined. The sample size S is chosen so as to minimize M subject to the constraint that the average number and the standard deviation of the number of overflow ranges should be less than 1. (In reality, the resulting value of M is not the minimum possible, but very nearly so.)

All the formulas in this section will be derived in the full version of the paper. The values of M and S are related by the formula

$$S = \frac{M - 2B}{K + 3P} \quad , \tag{3-1}$$

where B is the size of the input buffer. The values of M and S can be computed by solving two equations in two unknowns, M and r, where r is defined to be $F(S+1)/(N+1)$. The first equation guarantees that the above objective will be satisfied:

$$r - \ln r = \ln \left(\frac{(N+1)(R+P)}{M - 2(B+B')} \right) \quad , \tag{3-2}$$

where B and B' are the sizes of the small input and large output buffers, respectively, and P is the number of bytes per pointer field. The second equation

$$(M - 2B + K + 3P)(M - 2(B+B')) = r(N+1)(R+P)(K+3P) \quad , \tag{3-3}$$

specifies that the S sampled keys can be sorted internally. The value of M obtained by solving these two equations is given in Section 6 for two large-scale sorting examples. We have

$$M \cong \sqrt{\left(\ln\left(\frac{NR}{K}\right) + \ln\left(\frac{\ln(NR/K)}{2}\right) \right) \frac{NRK}{2}} + 2B' \quad .$$

We will show in the full paper that the standard deviation of the number of overflow buckets is less than 1. The average number of records in each bucket is at least $(1 - 1/r)F$, so the buckets tend to be nearly full.

4. BUBBLE MEMORY AS AN ASSOCIATIVE SECONDARY STORE

The basic requirement for the associative secondary storage device is that it must be able to process the following *associative search* (or *range query*), in about one or two seconds:

Given values a and b,
retrieve all records in the range (4-1)
$\{a < (\text{key value}) \leq b\}$.

This type of associative search is performed once for each bucket range during the Internal Sort Phase of BucketSort, and accounts for the great majority of the execution time. In the full paper, we show how to implement the associative secondary storage by use of magnetic bubble memory (MBM). The use of bubble memory for associative secondary storage in database applications is discussed in [Chang and Lee 75], [Chang 78], and [Doty et al 80].

The MBM secondary storage device consists a hierarchy of of several units. For our purposes, we shall picture each box as consisting of several thousand arrays, like the one pictured in Figure 1. Each array contains one megabit of storage. There is some random access memory (RAM) that serves as a read/write buffer for several of the arrays (perhaps one RAM per module). There are roughly 2000 bits of RAM devoted to each one-megabit array. An associative search is carried out by transferring each record from the array to the RAM; a microprocessor at the RAM selects and transfers to the bus all the records whose key values satisfy the range query (4-1).

The paths that connect the array to the RAM operate at one megabit/second, so assuming that the array can emit data at that speed from the minor loops to the read shift register (which can be shown to be true), the entire contents of the array can be searched in one second. All the arrays in all the boxes are searched *in parallel*, so the total time per associative search is one second. More details are given in the full paper.

5. SPEEDING UP THE BUCKET RETRIEVALS

In the Internal Sort Phase, an associative search as in (4-1) is performed to retrieve each bucket. The search done in the naive way described above requires about 1-2 seconds. The channel that connects the MBM to the CPU may be able to transfer M bytes (one internal memory load) in only a *fraction* of a second. In that case, the bottleneck is the search time. This section is devoted to software marking techniques that remove this bottleneck by speeding up the time per bucket retrieval, so that the rate of a bucket retrieval matches the channel speed.

Roughly one out of every NR/M records in the file belongs to a given bucket, so almost all the bucket retrieval time involves processing records that do *not* satisfy the range query (4-1). The key to speeding up the bucket retrievals is to avoid having to process most of these records. To accomplish that, we need one extra bit per record, which we store in the MBM and use as a *mark bit*. Records may be either marked or unmarked. The MBM architecture must be modified to support a special reading mode in which *only marked records are transferred from the major loops to the RSR*. Each range query now has the form

Given values a and b,
retrieve all marked records in the range (5-1)
$\{a < (\text{key value}) \leq b\}$.

The following simple example illustrates how we can reduce the time per bucket retrieval by a factor of k, for any $1 \leq k << NR / M$. A detailed analysis is given at the end of the section. We assume that during the combination subphase of the Bucket Formation Phase, we have stored the $k-1$ ordered key values that partition the buckets into k roughly equal-sized groups. We shall denote the $k-1$ ordered keys plus the dummy keys $-\infty$ and $+\infty$ by

$$-\infty = Y_0,\ Y_1,\ Y_2,\ \ldots,\ Y_{k-1}, Y_k = +\infty\ .$$

The Internal Sort Phase can be divided into k sequential subphases. The i^{th} subphase involves

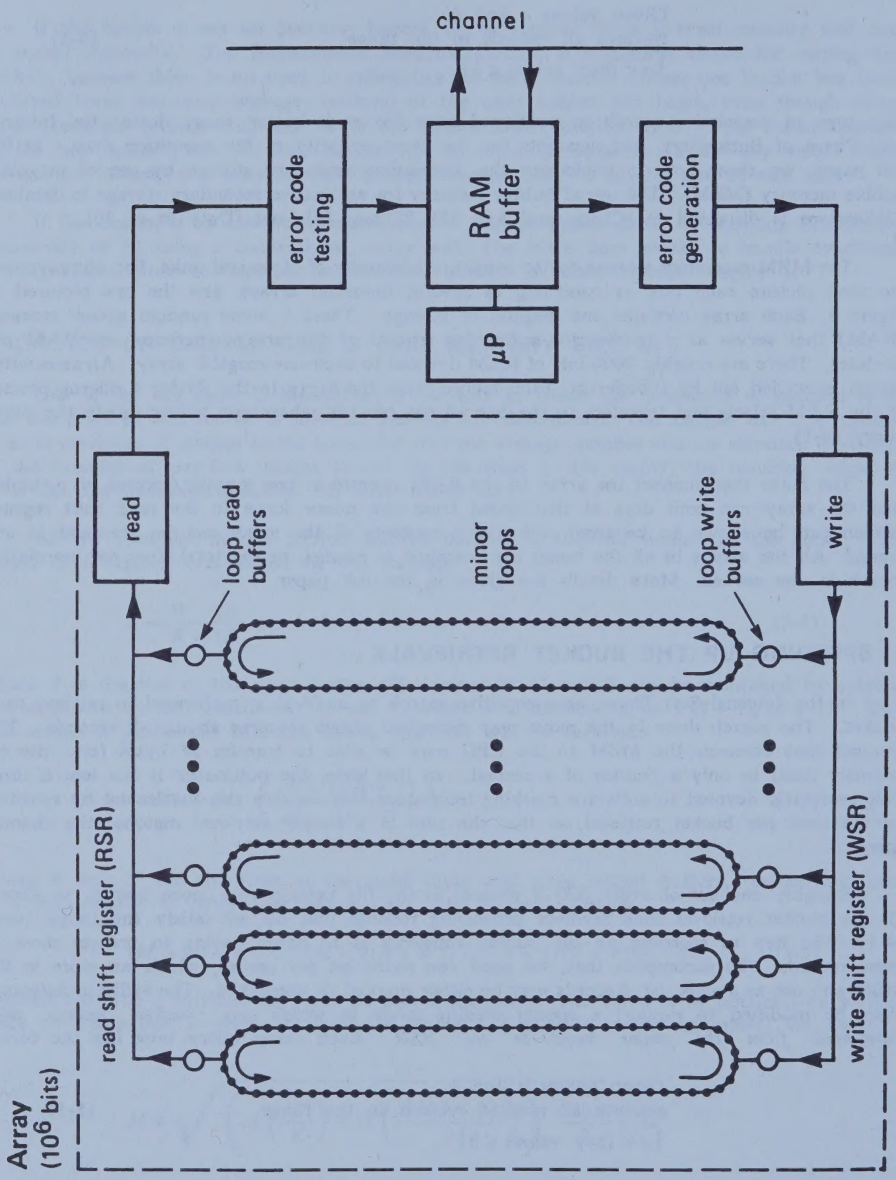


Figure 1

The basic structural unit of MBM is the array, enclosed by the dashed lines. The one-megabit array consists of 1000 minor loops, each loop containing 1000 rotating bits (magnetic bubbles).

the retrieval and sorting of all the buckets in the range between Y_{i-1} and Y_i. The new twist is that we do the following preprocessing at the beginning of the i^{th} subphase, for $i = 1, 2, ..., k$:

Mark all the records in the range
$\{Y_{i-1} < (\text{key value}) \leq Y_i\}$ (5-2)
and unmark the others.

Roughly N/k records are marked at any one time, so the number of processed records in each bucket retrieval decreases by a factor of k. Each of the k preprocessing steps (5-1) requires one complete associative search (which is 1-2 seconds), but as long as k is not too large, the total preprocessing time is negligible.

We can do even better if during each bucket retrieval we turn off the mark bits of all the records that satisfy the range query. At the beginning of each subphase, roughly N/k records are marked, but the number of marked records decreases linearly until it becomes 0 at the end of the subphase. The average number of records that are processed for each bucket retrieval is now approximately $N/(2k)$.

Modifications of the Marking Technique

The range query (5-1) can be done with only one comparison. Each preprocessing step (5-2) can also be done with only one comparison in a naive way that requires two passes over the file. It can be done in only one pass if two mark bits, M_1 and M_2, are used for each record. The details are covered in the full paper. In addition, we have developed techniques based on the fast random access of MBM to further reduce retrieval time.

Analysis of the Marking Technique

The rest of this section is devoted to a more realistic assessment of the marking technique. Details of the analysis will be given in the full paper. If we do the preprocessing as in (5-1) and (5-2) using k subphases, the time to access the records in a bucket will be reduced by some factor *less* than k. The net effect is that the number k of subphases should be increased slightly to have the desired effect.

First, the time for a bucket retrieval is limited by the channel speed. If the channel is at full capacity, the RSRs in some of the bubble devices may have to stop and wait for buffer space in the RAM to be cleared. When the channel becomes available and clears a buffer, there may be a slight delay before another record is processed that satisfies the range query. Extensive simulations for the two examples in the next section show that this effect is negligible. The channels remain 98%-99% busy. Details are given in the full paper.

The second complication deals with the fact that the arrays are searched in parallel during a bucket retrieval, and that some arrays take longer to search (i.e., they contain more marked records) than others. The total search time is equal to the *maximum* search time among all the arrays. If the number of subphases is k, for some $k \geq 1$, then on the average, $1000/k$ records in each array are marked at the start of each subphase. The standard deviation of the number of marked records is roughly $\sqrt{1000(k-1)/k^2}$. By the theory of statistics of extreme values ([Gumbel 58]), the average number of marked records in the array that contains the *most* is roughly $1000/k + \ell\sqrt{1000(k-1)/k^2}$, where the value of ℓ is $\sqrt{2\ln(0.4N/1000)}$. For example, if we have $N = 10^7$ and $k = 5$, then the largest number of marked records per array is approximately 254, on the average, rather than 1000/5 = 200. Hence, the search time for the first bucket in each subphase is reduced by a factor of 4 instead of 5. (The

search times for the remaining buckets in each subphase decrease linearly to 0.) Experiments confirm that these approximations are valid.

The third complication occurs when the physical records of the MBM are larger than the logical records in the file. The physical record size in the MBM is determined by the number of arrays that are packed together into one independent unit. For example, if 32 arrays are driven by the same magnetic field, then for all practical purposes we have one "super array," which is called a module, consisting of 32 x 1000 synchronized minor loops, each containing 1000 bits. The physical record size is 32 kilobits, which is four kilobytes.

Let us use $b \geq 1$ to denote the ratio of the physical record size to the logical record size. If the logical record size in a BucketSort application is one kilobyte in the above example, then $b = 4$ logical records must be packed into each physical record. The mark bits are associated with the physical records, not the logical records; there is one mark bit for each physical record. The physical record is marked when *any* of the logical records it contains should be "marked". If we use four subphases with the marking technique, then each subphase may have *all* the physical records marked, since each physical record might contain one logical record in each of the four quadrants! Choosing a larger number $k = k'b$ of subphases will guarantee that there will be at most N / k' marked records. On the average, roughly $k = -b / \ln(1 - 1 / k')$ subphases are necessary to produce an average of N / k' marked physical records.

6. TOTAL SORTING TIME

The following two examples represent two extremely large sorting applications. The first example illustrates the sizes of the sorts done once a night by major banks; the second example is the projected size of large sorts within this decade. The analysis is omitted for brevity.

Example 1	Example 2
$N = 2 \times 10^6$ records	$N = 10^7$ *records*
$R = 400$ bytes/record	$R = 10^3$ bytes/record
$K = 50$ bytes/key	$K = 10^2$ bytes/key

The buffer sizes in our examples are $B = R$ and $B' = 4.8 \times 10^4$ bytes; the latter was chosen equal to the track size of the IBM 3380 disk drive in order to reduce latency during the disk I/O in the Sample Phase. The pointer size $P = 3$ bytes allows full access to memory as long as $M \leq 1.6 \times 10^7$ bytes. We plug these parameters into formulas (3-2) and (3-3) and solve the resulting system of equations for M and r. The number of bytes of required memory is

$$M = \begin{cases} 7.2 \times 10^5 & \text{(example 1)} \\ 3.4 \times 10^6 & \text{(example 2)} \end{cases}$$

and the number of minibuckets per ideal memory load turns out to be

$$r = \begin{cases} 9.4 & \text{(example 1)} \\ 10.35 & \text{(example 2)} \end{cases}$$

The total sorting time in typical cases is dominated by the Internal Sort Phase. There are roughly $(r/(r-1))N/F$ buckets, and the time per bucket is approximately the I/O time FR/C; the total time is therefore roughly

$$\frac{r}{r-1} \quad \frac{NR}{C},$$

where C is the channel speed. For these examples, we assume that $C = 3$ megabytes/second. If the input file already resides on the MBM, the total sorting time is roughly five minutes (example 1) and one hour (example 2). If the input file must first be transferred to the MBM at three megabytes/second, the total sorting time increases to 10 minutes (example 1) and two hours (example 2). This represents a significant improvement over the optimized merge-based sorting methods that use disk storage, which require about one hour (example 1) and 10-15 hours (example 2) using the same amount of internal memory as BucketSort.

7. TAKING ADVANTAGE OF A VECTOR PROCESSOR

If the time T per associative probe and the final output time P for each bucket are small enough, then the running time of the Internal Sort Phase is dominated by the CPU time. Sorting can be made much faster by using the modification in which the internal memory is approximately doubled and partitioned into two halves. The amount of internal memory required for the two examples in Section 6 is 1.2 megabytes and 6.6 megabytes. Less memory can be used at the expense of more iterations in the Internal Sort Phase.

If the modification is used, then the internal sorting of each bucket is static rather than dynamic, so in place of Replacement Selection we can use static internal sorting methods like quicksort and radix sort, which are roughly twice as fast. These methods can be improved further by a factor of approximately 2-3 if the CPU has certain vector features, like vector chaining and the instructions "Compress into Vector Register" and "Shift Words out of Vector Register" ([Kulsrud et al 78], [Sedgewick 78]). Vector chaining is a particularly useful feature. Details are given in the full paper.

8. APPLICATIONS TO DATABASE SYSTEMS

Currently, large databases are stored on disk in a random access manner and use hashing and index techniques to speed up search time. We can instead store databases in bubble memory and take advantage of the faster random access. The same hashing and index techniques can be used. Such a database system is able to process transactions at a much faster rate than do disk systems.

An even bigger advantage of such a storage structure is that the database can perform general relational queries quickly (on the order of seconds) by making associative searches. This cannot be done when disk storage is used. In other words, the bubble memory can be used either as a random access device or as an associative device, whichever is more efficient for the current transaction or query. The mark bits used by BucketSort to speed up the bucket retrieval in the Internal Sort Phase, as described in Section 5, are also very helpful for performing relational queries ([Chang and Lee 78], [Shaw 80]). BucketSort can be viewed as one of the operations of the relational database system, which makes for a very powerful database environment.

9. CONCLUSIONS

We have shown that BucketSort is an extremely fast sorting method for very large files. In our average-case analysis, we make no assumptions about the distribution of the key values or about the initial order of the keys. There are no worst-case input files. For the two examples in Section 6, conservative estimates show that BucketSort is faster than optimized merge-based methods that use disk storage by a factor of roughly 10. The space and time efficiency can be improved by one of several modifications, as well as by the use of faster I/O channels or vector processors. BucketSort fits naturally in relational database systems.

One contribution of BucketSort is the sampling technique used to partition the file into roughly equal-sized bucket ranges. This technique should prove useful in other sorting and searching applications in which files must be processed one memory load at a time, such as in parallel sorting and in space-constrained combinatorial algorithms. We have shown that the associative secondary storage device can be realized effectively by using magnetic bubble memory (MBM), which can perform an associative search in 1-2 seconds. The software marking techniques discussed in Section 5 can reduce the search time for this application to a fraction of a second.

10. REFERENCES

1. H. Chang. "On Bubble Memories and Relational Data Base." *Proceedings of the Fourth International Conference on Very Large Databases*, September 1978.

2. H. Chang and S. Y. Lee. "Associative-Search Bubble Devices for Content-Addressable Memories." *IBM Technical Disclosure Bulletin 18,* 598-602, July 1975.

3. K. L. Doty, J. D. Greenblatt, and S. Y. N. Su. "Magnetic Bubble Memory Architecture for Supporting Associative Searching of Relational Databases." *IEEE Transactions on Computers C-29* (11), 951-970, November 1980.

4. E. J. Gumbel. *Statistics of Extremes.* Columbia University Press, New York, 1958.

5. D. L. Johnson and M. J. Nishimura. "Associative Secondary Memories in Large External Sorts: Radix Sort, BucketSort, and MaxiBucketSort Developments." Univ. of Washington.

6. D. E. Knuth. *The Art of Computer Programming,* Volume 2: *Seminumerical Algorithms.* Addison-Wesley, Reading, Massachusetts, second edition 1981.

7. H. E. Kulsrud, R. Sedgewick, P. Smith, and T. Szymanski. "Partition Sorting on the CRAY-1." SCAMP Working Paper No. 7/78, Institute for Defense Analyses, Princeton, N.J., September 1978.

8. C. S. Lin. "Sorting with Associative Secondary Storage Devices." *Proceedings of the AFIPS National Computer Conference,* 691-695, 1977.

9. E. E. Lindstrom and J. S. Vitter. "The Design and Analysis of BucketSort for Bubble Memory Secondary Storage." *IEEE Transactions on Computers*, to appear.

10. M. Rabin. "Probabilistic Algorithms," in *Algorithms and Complexity*, ed. by J. Traub, Academic Press, New York, 21-40, 1976.

11. R. Sedgewick. "Sorting on the CRAY-1: An Overview," SCAMP Working Paper No. 16/78, Institute for Defense Analyses, Princeton, NJ, September 1978.

12. D. E. Shaw. "A Relational Database Machine Architecture," *Proceedings of the Fifth ACM Workshop on Computer Architectures for Non-Numeric Processing*, Pacfic Grove, CA, 84-95, March 1980.

13. J. S. Vitter. "Faster Methods for Random Sampling." *Communications of the ACM* 27(7), 703-718, July 1984.

UNDETECTED ERROR PROBABILITY AND THROUGHPUT ANALYSIS OF A CONCATENATED CODING SCHEME

HUIJIE DENG and DANIEL J. COSTELLO, JR.

Department of Electrical & Computer Engineering
Illinois Institute of Technology
Chicago, IL 60616

ABSTRACT

The performance of a proposed concatenated coding scheme for error control on a NASA telecommand system is analyzed. In this scheme, the inner code is a distance-4 Hamming code used for both error correction and error detection. The outer code is a shortened distance-4 Hamming code used only for error detection. Interleaving is assumed between the inner and outer codes. A retransmission is requested if either the inner or outer code detects the presence of errors. Both the undetected error probability and the throughput of the system are analyzed. Results indicate that high throughputs and extremely low undetected error probabilities are achievable using this scheme.

I. INTRODUCTION

Consider a concatenated coding scheme for error control on a binary symmetric channel, called the inner channel. The bit error rate (BER) of the channel is called the inner BER, and is denoted by ε_i. Two linear block codes, C_f and C_b, are used. The inner code C_f, called the frame code, is an (n,k) systematic binary block code with minimum distance d_f. The frame code is designed to correct t or fewer errors and simultaneously detect $\lambda(\lambda > t)$ or fewer errors, where $t+\lambda+1 \leq d_f$ [1]. The outer code is an (n_b, k_b) binary block code with

$$n_b = mk, \tag{1}$$

where m, a positive integer, is the number of frames. The outer code is designed for error detection only.

The encoding of the concatenated code is achieved in two stages (see Figure 1). A message of k_b bits is first encoded into a codeword of n_b bits in the outer code C_b. Then this codeword is interleaved to depth m. After interleaving, the n_b-bit block is divided into m k-bit segments. Each k-bit segment is encoded into an n-bit word in the frame code C_f. This n-bit word is called a frame. The two dimensional block format is depicted in Figure 2.

The decoding consists of error correction and error detection on each frame and error detection on the m decoded k-bit segments. When a frame in a block is received, it is first decoded based on the frame code C_f. The n-k parity bits are then removed from the decoded frame. If there are t or fewer transmission errors in a received frame, the errors will be corrected, and the decoded segment is error free. If there are more than t errors in the received frame, the errors will be either detected or undetected. If the errors are detected, the decoder stops decoding immediately and requests a retransmission of the entire block. On the other hand, if the errors in a frame are undetected, the decoded segment will be stored in a buffer and the decoder continues to decode the next frame. After m frames of a block have been decoded, the m k-bit decoded segments are then deinterleaved. Error detection is performed on these deinterleaved m segments based on the outer code C_b. If no errors are detected, the m decoded segments are assumed to be error free, and are accepted by the receiver. If the presence of errors

is detected, the m decoded segments are discarded and the receiver requests a retransmission of the entire block.

The error control scheme described above is actually a combination of forward-error-correction (FEC) and automatic-repeat-request (ARQ), called a hybrid ARQ scheme [1]. In this paper, we analyze the performance of the proposed error control scheme. Specifically, the system reliability and the system throughput are calculated. The system reliability is measured in terms of the probability of undetected error after decoding. The system throughput is determined by the retransmission strategy, which may be one of three basic modes, namely, stop-and-wait, go-back-N, or selective-repeat. Only the selective-repeat retransmission strategy is considered in this paper. A particular scheme, proposed for a NASA telecommand system, is analyzed as an example.

II. PROBABILITY OF UNDETECTED ERROR FOR THE INNER CODE

Let $P_c^{(f)}(\varepsilon_i)$ denote the probability of correct decoding for the frame code. Suppose that a bounded-distance decoding algorithm is employed. Bounded-distance decoding corrects all received n-bit sequences with t or fewer errors. When an n-bit sequence with more than t errors is detected, no attempt is made to correct the errors. Since there are $\binom{n}{i}$ distinct ways in which i errors may occur among n bits,

$$P_c^{(f)}(\varepsilon_i) = \sum_{i=0}^{t} \binom{n}{i} \varepsilon_i^{\,i} (1-\varepsilon_i)^{n-i} \tag{2}$$

for bounded-distance decoding.

For a code word $\bar{v}$ in the frame code C_f, let $w(\bar{v})$ denote the Hamming weight of $\bar{v}$. If a decoded frame contains an undetectable error pattern, this error pattern must be a nonzero codeword in C_f. Let $\bar{e}_0$ be a nonzero error pattern after decoding. The probability $P_f(w, \varepsilon_i)$ that a decoded frame contains a nonzero error pattern $\bar{e}_0$ after decoding is given by [2-4]

$$P_f(w, \varepsilon_i) = \sum_{i=0}^{t} \sum_{j=0}^{\min(t-i,n-w)} \binom{w}{i} \binom{n-w}{j} \varepsilon_i^{\,w-i+j} (1-\varepsilon_i)^{n-w+i-j}, \tag{3}$$

where $w = w(\bar{e}_0)$, and ε_i is the BER of the inner channel. If $\varepsilon_i \ll \frac{1}{n}$, then

$$P_f(w, \varepsilon_i) \approx \binom{w}{t} \varepsilon_i^{\,w-t} (1-\varepsilon_i)^{n-w+t}. \tag{4}$$

Let $P_{ud}^{(f)}(\varepsilon_i)$ denote the probability of undetected error for the frame code. Let $\{A_w^{(f)}, d_f \le w \le n\}$ be the weight distribution of C_f. It follows from (3) and (4) that

$$P_{ud}^{(f)}(\varepsilon_i) = \sum_{w=d_f}^{n} A_w^{(f)} P_f(w, \varepsilon_i) \tag{5}$$

$$P_{ud}^{(f)}(\varepsilon_i) \approx A_{d_f}^{(f)} P_f(d_f, \varepsilon_i)$$

$$\approx A_{d_f}^{(f)} \binom{d_f}{t} \varepsilon_i^{\,d_f-t} (1-\varepsilon_i)^{n-d_f+t}, \quad \text{for } \varepsilon_i \ll \frac{1}{n}. \tag{6}$$

Now consider any one of the m frames, say the j^{th} frame. If the decoded frame contains undetected errors, the BER ε_a after decoding is given by

$$\varepsilon_a = \frac{1}{n} \sum_{w=d_f}^{n} A_w^{(f)} w \, P_f(w, \varepsilon_i). \tag{7}$$

For $\varepsilon_i << \frac{1}{n}$, then

$$\varepsilon_a \approx \frac{1}{n} d_f A_{d_f}^{(f)} P_f(d_f, \varepsilon_i) \approx \frac{d_f}{n} A_{d_f}^{(f)} \binom{d_f}{t} \varepsilon_i^{d_f - t} (1-\varepsilon_i)^{n-d_f+t}, \tag{8}$$

will be a good approximation to ε_a. Let E be defined as the event that a frame contains undetected errors. Now let $\varepsilon_{a/E}$ denote the BER embedded in a decoded frame conditioned on the occurrence of event E. It follows from (7) that

$$\varepsilon_{a/E} = \varepsilon_a / Pr\{E\} \tag{9}$$

$$= \varepsilon_a / P_{ud}^{(f)}(\varepsilon_i). \tag{10}$$

For $\varepsilon_i << \frac{1}{n}$, substituting (6) and (8) into (10) yields

$$\varepsilon_{a/E} \approx \frac{\frac{1}{n} d_f A_{d_f}^{(f)} P_f(d_f, \varepsilon_i)}{A_{d_f}^{(f)} P_f(d_f, \varepsilon_i)} = \frac{d_f}{n}. \tag{11}$$

Now define S to be a random variable such that when h of the m frames contain undetected errors, and the remaining m-h frames are decoded correctly, S = h, h = 0, 1, 2, ..., m. It follows from (2) and (5) that

$$Pr\{S=h\} = \binom{m}{h} \cdot [P_{ud}^{(f)}(\varepsilon_i)]^h \cdot [P_c^{(f)}(\varepsilon_i)]^{m-h}. \tag{12}$$

Note that (12) is not a binomial distribution because $P_{ud}^{(f)}(\varepsilon_i) + P_c^{(f)}(\varepsilon_i) < 1$, i.e., some received sequences with more than t errors are detected by the frame code.

After deinterleaving of the m decoded segments (with the n-k parity bits removed from each frame), the BER embedded in the n_b-bit block, conditioned on S=h, is given by

$$\varepsilon_0(h) = \varepsilon_{a/E} \cdot \frac{h}{m}, \quad h = 0, 1, 2, \ldots, m. \tag{13}$$

We call the channel specified by (12) and (13) the outer channel, and it is depicted in Figure 3. Note that $\varepsilon_0(0) = 0$. This channel can be viewed as a block interference (BI) channel, as described in [5]. Δ_h, h = 0, 1, ..., m, is called the h-th component channel of the BI channel. Each block of n_b bits (n_b is the length of the outer code) is transmitted over one of the m component channels. The random variable S determines which component channel is used to transmit a given n_b-bit block.

III. PERFORMANCE OF THE CONCATENATED CODE

Let $\{A_i^{(b)}, d_b \le i \le n_b\}$ be the weight distribution of the outer code, where d_b is the minimum distance of C_b. Let $P_{ud}^{(b)}(\varepsilon)$ be the probability of undetected error for the outer code C_b. If the n_b-bit block is transmitted over the h-th component channel Δ_h of the outer channel, it follows from (13) that

$$P_{ud}^{(b)}(\varepsilon_0(h)) = \sum_{i=d_b}^{n_b} A_i^{(b)} (\varepsilon_0(h))^i (1-\varepsilon_0(h))^{n_b - i}. \tag{14}$$

Let $P_{ud}(\varepsilon_i)$ be the average probability of undetected error of the con-

catenated code. From (12) and (14) we obtain

$$P_{ud}(\varepsilon_i) = \sum_{h=0}^{m} \Pr\{S=h\} P_{ud}^{(b)}(\varepsilon_0(h))$$

$$= \sum_{h=1}^{m} \{\binom{m}{h}[P_{ud}^{(f)}(\varepsilon_i)]^h [P_c^{(f)}(\varepsilon_i)]^{m-h} \cdot \sum_{i=d_b}^{n_b} A_i^{(b)} (\varepsilon_0(h))^i (1-\varepsilon_0(h))^{n_b-i}\}, \quad (15)$$

where $P_c^{(f)}(\varepsilon_i)$ and $P_{ud}^{(f)}(\varepsilon_i)$ are given by (2) and (5), respectively.

Suppose that the selective-repeat ARQ scheme is used as the retransmission strategy. The specific manner in which the receiver signals to the transmitter for a retransmission will not be considered. It will be assumed, however, that this backward signal is error-free, and that repeated retransmissions of a block are possible. Thus for the concatenated code, let $P_{ud}(\varepsilon_i)$, $P_r(\varepsilon_i)$, and $P_c(\varepsilon_i)$ denote the probabilities of an undetected error, of a block retransmission, and of correct decoding, respectively. Obviously,

$$P_{ud}(\varepsilon_i) + P_r(\varepsilon_i) + P_c(\varepsilon_i) = 1. \quad (16)$$

Then the throughput of the system is [1]

$$\eta = \frac{k}{n} \cdot \frac{k_b}{n_b} (1-P_r(\varepsilon_i)) = \frac{k}{n} \cdot \frac{k_b}{n_b} \cdot (P_{ud}(\varepsilon_i) + P_c(\varepsilon_i)). \quad (17)$$

Note that a transmitted block will be received correctly if and only if all m frames are decoded correctly. Therefore, the probability of accepting a correct block is given by

$$P_c(\varepsilon_i) = [P_c^{(f)}(\varepsilon_i)]^m = [\sum_{i=0}^{t} \binom{n}{i} \varepsilon_i^i (1-\varepsilon_i)^{n-i}]^m. \quad (18)$$

Because $P_{ud}(\varepsilon_i) \ll P_c(\varepsilon_i)$, it follows from (17) and (18) that

$$\eta \approx \frac{k}{n} \cdot \frac{k_b}{n_b} \cdot [\sum_{i=0}^{t} \binom{n}{i} \varepsilon_i^i (1-\varepsilon_i)^{n-i}]^m. \quad (19)$$

It can easily be seen that η increases monotonically as t increases, but that for small ε_i, η is only a weakly increasing function of t.

In order to see the relationship between t and $P_{ud}(\varepsilon_i)$, from (15) we have

$$P_{ud}(\varepsilon_i) \approx m \cdot P_{ud}^{(f)}(\varepsilon_i) \cdot [P_c^{(f)}(\varepsilon_i)]^{m-1} \cdot \{\sum_{i=d_b}^{n_b} A_i^{(b)} (\varepsilon_0(1))^i (1-\varepsilon_0(1))^{n_b-i}\},$$

$$\text{for} \quad \varepsilon_i \ll \frac{1}{n}. \quad (20)$$

Using (6), (11), and (13), $P_{ud}(\varepsilon_i)$ can be further approximated as

$$P_{ud}(\varepsilon_i) \approx K \cdot \binom{d_f}{t} \varepsilon_i^{d_f-t} (1-\varepsilon_i)^{n-d_f+t} [P_c^{(f)}(\varepsilon_i)]^{m-1}, \quad (21)$$

where

$$K = m \cdot A_{d_f}^{(f)} \{ \sum_{i=d_b}^{n_b} A_i^{(b)} \left(\frac{d_f}{m \cdot n} \right)^i \left(1 - \frac{d_f}{m \cdot n} \right)^{n_b - i} \}$$

is a constant which is independent of t. Let Q(t) denote the right hand side of (21),

$$\frac{Q(t+1)}{Q(t)} \approx \frac{(d_f - t)}{(t+1)} \cdot \frac{1}{\varepsilon_i} >> n, \quad \text{for} \quad \varepsilon_i << \frac{1}{n} . \tag{22}$$

That is, for $\varepsilon_i << 1/n$, when t increases by 1, $P_{ud}(\varepsilon_i)$, the probability of undetected error, will increase by approximately ε_i^{-1}. Thus $P_{ud}(\varepsilon_i)$ is a strongly increasing function of t. For this reason, a large value of t is not desirable in such a system.

IV. EXAMPLE

We now consider a particular scheme proposed for a NASA telecommand system. The frame code C_f is a distance-4 Hamming code with generator polynomial

$$g(x) = (x+1)(x^6+x+1) = x^7 + x^6 + x^2 + 1,$$

where $x^6 + x+1$ is a primitive polynomial of degree 6. The natural length of this code is 63. This code is used for single error correction, and is also used to detect all error patterns of double weight and some higher odd weight error patterns. The outer code is a distance-4 shortened Hamming code with generator polynomial

$$\begin{aligned} g(x) &= (x+1)(x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x+1) \\ &= x^{16} + x^{12} + x^5 + 1, \end{aligned}$$

where $x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x+1$ is a primitive polynomial of degree 15. This code is the X.25 standard for packet-switched data networks [6]. The natural length of this code is $2^{15} - 1 = 32,767$. In this example, a shortened code of maximum length 3,584 bits is considered. This code is used for error detection only.

We assume that the number of information bytes in a frame is between 3 and 7, that is, the inner code can also be shortened. The number of frames in a block is between 4 and 64.

To obtain a precise result for $P_{ud}(\varepsilon_i)$, a computer program was written to help determine the reliability of the proposed concatenated coding scheme. We found that if only one frame contains a weight 4 undetected error pattern, then this error pattern can always be detected by the outer code. Thus (15) can be modified as follows:

$$\begin{aligned} P_{ud}(\varepsilon_i) = {} & \binom{m}{1} \overline{P}_{ud}^{(f)}(\varepsilon_i) P_c^{(f)}(\varepsilon_i)^{m-1} \\ & \cdot \sum_{i=d_b}^{n_b} A_i^{(b)} (\overline{\varepsilon}_0(1))^i (1-\overline{\varepsilon}_0(1))^{n_b - i} \\ & + \sum_{h=2}^{m} \{ \binom{m}{h} [P_{ud}^{(f)}(\varepsilon_i)]^h [P_c^{(f)}(\varepsilon_i)]^{m-h} \\ & \cdot \sum_{i=d_b}^{n_b} A_i^{(b)} (\varepsilon_0(h))^i (1-\varepsilon_0(h))^{n_b - i} \}, \end{aligned} \tag{23}$$

where

$$\overline{P}_{ud}^{(f)}(\varepsilon_i) = \sum_{w=d_f+1}^{n} A_w^{(f)} P_f(w, \varepsilon_i), \tag{24}$$

and

$$\overline{\varepsilon}_0(1) = \frac{\frac{1}{n}\sum_{w=d_f+1}^{n} A_w^{(f)} w\, P_f(w, \varepsilon_i)}{\overline{P}_{ud}^{(f)}(\varepsilon_i)} \cdot \frac{1}{m}. \tag{25}$$

The weight distributions of the frame code are listed in Table 1. Results for $P_{ud}(\varepsilon_i)$, based on (23) for various values of ε_i, are given in Table 2, where we have used the method in [7] to obtain

$$P_{ud}^{(b)}(\varepsilon_0(h)) = \sum_{i=d_b}^{n_b} A_i^{(b)} (\varepsilon_0(h))^i (1-\varepsilon_0(h))^{n_b-i}.$$

$1-P_c(\varepsilon_i)$, the probability of not decoding correctly, and η, the system throughput, are given in Tables 3 and 4, respectively, for $\varepsilon_i = 10^{-4}$. Clearly, the proposed system is capable of achieving high throughput and low undetected error probability over a wide range of system parameters.

REFERENCES

[1] S. Lin and D.J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, New Jersey, 1983.

[2] E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.

[3] J. MacWilliams, "A Theorem on the Distribution of Weights in a Systematic Code", Bell Sys. Tech. J., Vol. 42, pp. 79-94, 1963.

[4] Z. McHuntoon and A.M. Michelson, "On the Computation of the Probability of Post-Decoding Error Events for Block Codes", IEEE Trans. on Inform. Theory, IT-23, No. 3, May 1977, pp. 399-403.

[5] R.J. McEliece and W.E. Stark, "Channels with Block Interference", IEEE Trans. on Inform. Theory, IT-30, pp. 44-53, Jan. 1984.

[6] CCITT: Recommendation X.25, "Interface Between Data Terminal Equipment for Terminals Operating in Packet Mode on Public Data Networks", with Plenary Assembly, Doc. No. 7, Geneva, 1980.

[7] T. Fujiwara, T. Kasami, A. Kitai, and S. Lin, "On the Undetected Error Probability for Shortened Hamming Codes", Technical Report, NASA Grant NAG 5-234, August 1983.

ACKNOWLEDGEMENT

This work was supported by NASA under Grant NAG 5-234.

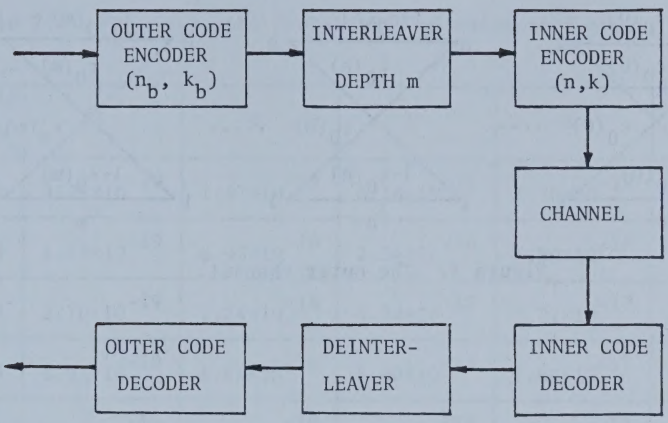


Figure 1. A concatenated coding system.

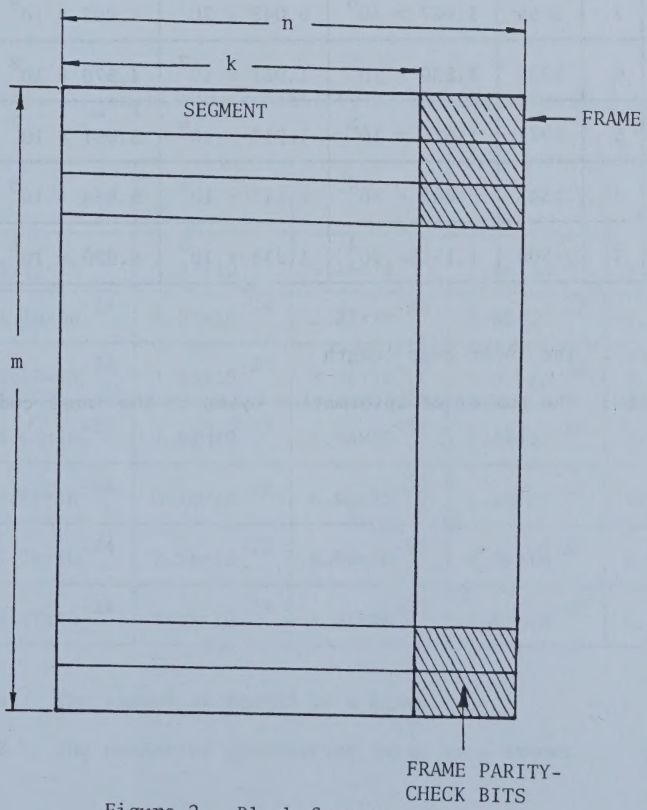


Figure 2. Block format.

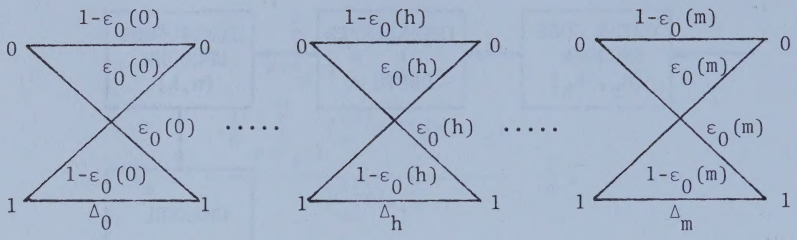


Figure 3. The outer channel.

Table 1. The inner code weight distribution.

n	IB	$A_4^{(f)}$	$A_6^{(f)}$	$A_8^{(f)}$	$A_{10}^{(f)}$
63	7	9765	1.057×10^6	6.048×10^7	1.997×10^9
55	6	5592	4.530×10^5	1.902×10^7	4.570×10^8
47	5	2927	1.678×10^5	4.913×10^6	8.091×10^7
39	4	1343	5.098×10^4	9.613×10^5	9.934×10^6
31	3	505	1.151×10^4	1.233×10^5	6.930×10^5

n : The inner code length

IB : The number of information bytes in the inner code

Table 2(A). $P_{ud}(\varepsilon_i)$, the probability of undetected error, for $\varepsilon_i=10^{-4}$.

m \ IB	3	4	5	6	7
4	4.25×10^{-20}	1.87×10^{-19}	6.14×10^{-19}	1.70×10^{-18}	4.01×10^{-18}
14	1.53×10^{-19}	6.93×10^{-19}	2.34×10^{-18}	6.50×10^{-18}	1.56×10^{-17}
24	2.70×10^{-19}	1.24×10^{-18}	4.24×10^{-18}	1.21×10^{-17}	2.97×10^{-17}
34	3.93×10^{-19}	1.83×10^{-18}	6.40×10^{-18}	1.84×10^{-17}	4.60×10^{-17}
44	5.22×10^{-19}	2.47×10^{-18}	8.74×10^{-18}	2.55×10^{-17}	6.46×10^{-17}
54	6.58×10^{-19}	3.15×10^{-18}	1.13×10^{-17}	3.33×10^{-17}	8.56×10^{-17}
64	8.00×10^{-19}	3.87×10^{-18}	1.41×10^{-17}	4.19×10^{-17}	1.09×10^{-16}

Table 2(B). $P_{ud}(\varepsilon_i)$, the probability of undetected error, for $\varepsilon_i=10^{-5}$.

m \ IB	3	4	5	6	7
4	4.23×10^{-25}	1.87×10^{-24}	6.16×10^{-24}	1.66×10^{-23}	3.89×10^{-23}
14	1.48×10^{-24}	6.57×10^{-24}	2.21×10^{-23}	5.88×10^{-23}	1.38×10^{-22}
24	2.55×10^{-24}	1.13×10^{-23}	3.74×10^{-23}	1.02×10^{-22}	2.39×10^{-22}
34	3.62×10^{-24}	1.62×10^{-23}	5.34×10^{-23}	1.45×10^{-22}	3.42×10^{-22}
44	4.69×10^{-24}	2.10×10^{-23}	6.96×10^{-23}	1.90×10^{-22}	4.48×10^{-22}
54	5.78×10^{-24}	2.58×10^{-23}	8.59×10^{-23}	2.39×10^{-22}	5.56×10^{-22}
64	6.87×10^{-24}	3.08×10^{-23}	1.03×10^{-22}	2.81×10^{-22}	6.22×10^{-22}

m : The number of frames in a block

IB : The number of information bytes in a frame.

Table 3. $1-P_c(\varepsilon_i)$, The probability of not decoding correctly, for $\varepsilon_i=10^{-4}$.

$1-P_c(\varepsilon_i)$ m \ IB	3	4	5	6	7
4	1.86×10^{-5}	2.96×10^{-5}	4.31×10^{-5}	5.92×10^{-5}	7.78×10^{-5}
14	6.50×10^{-5}	1.04×10^{-4}	1.51×10^{-4}	2.07×10^{-4}	2.72×10^{-4}
24	1.11×10^{-4}	1.77×10^{-4}	2.59×10^{-4}	3.55×10^{-4}	4.67×10^{-4}
34	1.58×10^{-4}	2.51×10^{-4}	3.66×10^{-4}	5.03×10^{-4}	6.61×10^{-4}
44	2.04×10^{-4}	3.25×10^{-4}	4.74×10^{-4}	6.51×10^{-4}	8.56×10^{-4}
54	2.51×10^{-4}	3.99×10^{-4}	5.82×10^{-4}	7.98×10^{-4}	1.05×10^{-3}
64	2.97×10^{-4}	4.73×10^{-4}	6.89×10^{-4}	9.46×10^{-4}	1.24×10^{-3}

Table 4. η, The system throughput, for $\varepsilon_i=10^{-4}$.

m \ IB	3	4	5	6	7
4	0.64515	0.71793	0.76591	0.79995	0.82533
14	0.73728	0.79113	0.82662	0.85177	0.87051
24	0.75260	0.80328	0.83666	0.86020	0.87790
34	0.75889	0.80824	0.84074	0.86360	0.88084
44	0.76231	0.81093	0.84293	0.86555	0.88236
54	0.76444	0.81259	0.84427	0.86665	0.88326
64	0.76590	0.81372	0.84494	0.86736	0.88382

m : The number of frames in a block

IB : The number of information bytes in a frame.

USING SIMULATED ANNEALING TO CALCULATE COMBINATORIAL CONSTANTS

*Lane A. Hemachandra**
Computer Science Department
Cornell University
Ithaca, New York 14853

Victor K. Wei
Bell Communications Research
Murray Hill, New Jersey 07974

Abstract

Simulated annealing is a heuristic procedure for solving optimization problems. We present a new application of simulated annealing: the calculation of combinatorical constants. For two classical problems, constant weight binary codes and spherical codes, we obtain new constructive lower bounds via simulated annealing.

1. Background

Annealing is the process of heating up and then cooling down materials. If the cooling is done slowly, the system of interacting atoms tends to settle in a low-energy state. A fundamental question in statistical mechanics is whether the atoms solidify and, if they do, whether they form a crystalline solid or a glass. Motivated by a subtlety of numerical integration, the Metropolis algorithm [Met53] has often been used to numerically simulate the annealing process to gain understanding of the ground-state configurations in the limit of extremely low temperatures.

Combinatorial optimization problems are concerned with finding the minimum of a given objective (or cost) function of many variables. The variables are subject to certain constraints, and they interact with each other in a way not unlike the atoms in a physical system. By appropriately defining an effective temperature for the multi-variable system and imitating the physical annealing process, researchers have sought to solve to a diverse collection of

*Work performed while with Bell Communications Research.

problems, with varying degrees of success.

Kirkpatrick [Kir83] first investigated the use of simulated annealing in connection with the physical design of computers. He found the technique of simulated annealing to be useful for solving layout and partitioning problems. More recently, researchers have compared simulated annealing with other heuristic techniques for *NP*-complete problems. At present, simulated annealing provides the best heuristic algorithm for the graph partitioning problem, but extensive fine-tuning has failed to make it the method of choice for graph coloring or the traveling salesman problem [Joh84][JohUN] [Gar79].

The successes of simulated annealing have resulted in a surge of interest in the method, and many evaluations of the technique and its applicability to diverse areas are in progress [Rom84][Sua84][Vec83][Whi84]. In this paper, we use simulated annealing to obtain improved lower bounds for certain combinatorial constants from classical coding theory. Our improvements are constructive, so the efficient codes found by simulated annealing can be used in practical applications.

2. The Simulated Annealing Procedure

Metropolis developed a procedure for generating a population of configurations from a large state space. Given a state, we perturb it by some small amount. If this perturbation has decreased the energy of the system ($\Delta E < 0$), we always switch to the new state. However, if the energy increases, we accept the new state with probability $\exp(-\Delta E/kT)$. Metropolis proved that in the limit of many state changes this procedure gives Boltzman's distribution [Met53].

The idea behind simulated annealing is as follows. We choose an energy (objective) function so that low energy corresponds to the type of state we seek. Using the Metropolis procedure should give the Boltzman distribution. Thus, at a low temperature, we find a state whose energy is near minimum (that of the ground state) with high probability.

In practice, computational costs dictate a modification of this procedure. In most studies, the problem is "cooled" in a series of stages, until it reaches a stable state. A simplified sketch of the procedure is shown in Figure 0. For each application, one must choose the *annealing schedule* (initial temperature, temperature decrements, and termination conditions), the *objective* (energy) *function*, and the *perturbation*.

Though there is some interest in the choice of annealing schedules [Lun83], annealing seems to be robust with respect to reasonable changes in schedule (provided the CPU time remains the same). Typically, the temperature is dropped exponentially, $T_i = \alpha^i T_0$, $0 < \alpha < 1$.

3. Two Classical Problems

3.1. Constant Weight Binary Codes

$A(n,d,w)$ is the maximum size of a binary code of length n and weight w with minimum distance at least d. A *binary code* of *length* n and *width* w is a set of binary n-tuples, each having w ones. The *Hamming distance* between two codewords is the number of bit positions in which they differ. The *minimum distance* of a code is the Hamming distance between its nearest pair of codewords. Codes can be used to correct transmission errors in data communications. It is desirable to have codes with large minimum distance or, equivalently, to have many codewords for a given length n, weight w, and minimum distance d. Know results for constant-weight binary codes can be found in [Mac77][Gra80][SloUNa].

3.2. Spherical Codes

$M(n,\theta)$ is the maximum cardinality of a set of vectors $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_k\}$ such that for each $\mathbf{x}_i \neq \mathbf{x}_j$, the L_2 norm $|\mathbf{x}_i|_2 = 1$ and the dot product $\mathbf{x}_i \cdot \mathbf{x}_j \leq \cos\theta$. Thus, $M(n,\theta)$ defines the maximum number of points on an n-dimensional spherical surface with minimum separating angle θ. Such a constellation of points is called a *spherical code*. Spherical codes

have important applications to transmission through Gaussian channels [Bla71][Bal75], and to many other areas [Ger79] [She80]. Known results on spherical codes can be found in [Cox62] and [Slo81].

4. Annealing Results

4.1. Constant Weight Binary Codes

For constant weight binary codes, we fix the number of codewords and try, by simulated annealing, to make the minimum distance as large as possible. Our perturbation is the simultaneous exchange of a zero position and a one position in a codeword.

We obtain the best results with a novel energy function. Since our goal is the maximize the minimum distance, we require the ordering on codes implicit in the energy function to correspond to a lexicographical ordering on the *distance profile*. (The i^{th} element of the distance profile of a code C $(dp_i(C))$ is the number of pairs of words in the code with Hamming distance exactly i.) This energy function favors increasing the minimum distance if possible, and otherwise decreases the number of pairs of words at just over the current minimum distance. Intuitively, it favors configurations that are more "spread out." Using the fact that Hamming distances in constant weight codes are always even, we implement this ordering as:

$$\text{energy}(C) = \sum_{i \geq 0} dp_i(C) \cdot \left(\binom{n}{2}+1\right)^{-i/2}$$

For our investigations, we concentrate on the distance 10 case, which corresponds to 5-error-detecting, 4-error-correcting codes. For eleven pairs of (n,w) (see figures 1 and 2) we find values better than the tables of [Gra80]*.

* However, recent results [SloUNb] tie or beat many of our values. To the authors' knowledge, for n=23, w=7 and n=24, w=8, our values are strictly better than any others.

4.2. Spherical Codes

For the case of spherical codes, in each run we fix the number of codewords and seek the greatest possible minimum separating angle. Our perturbation consists of changing the value of two coordinates of a codeword slightly, in such a way that the word remains on the unit hypersphere. The energy function is a common one, chosen by analogy with physical potential functions [Kir83]: $\text{energy}(C) = \sum_{i \neq j} \arccos(c_i \cdot c_j)^{-k}$, where k is an experimental parameter and we adopt the standard inner product.

We compare our results, in three dimensions, to those codes derived from the face-centered lattice [Slo81], and also with a recent lower bound [HemUN]. Figure 3 summarizes our results. Simulated annealing provides better codes than either method. Figure 4 provides an sample code. Preliminary investigations in four dimensions also look promising. Simulated annealing shows that $M(4, \arccos(5/6)) \geq 97$, which betters the lattice bound of 96, and $M(4,.504) \geq 145$, which betters the lattice bound of $M(4,\arccos(9/10)) \geq 144$.

5. Conclusions

We have used simulated annealing to derive new, constructive, lower bounds for two classes of combinatorial constants. Though a new technique, we feel simulated annealing will become a core technique not only in the design of heuristic algorithms for *NP*-complete problems, but also in the improvement of combinatorial constants.

References

[Bla71] I.F. Blake, "The Leech lattice as a code for the Gaussian channel," *Inform. Contr.*, vol. 19, pp. 66-74, 1971.

[Bla75] I.F. Blake and R.C. Mullin, *The Mathematical Theory of Coding*, New York, Academic Press, 1975.

[Cox62] H.S.M. Coxeter, "The problem of packing a number of equal nonoverlapping circles on a sphere," *Trans. New York Acad. Sci.*, series II, vol. 24, no. 3, pp. 320-331, Jan. 1962.

[Gar79] M.R. Garey and D.S. Johnson, *Computers and Intractability: a guide to the theory of NP-Completeness*,

Freeman, 1979.

[Ger79]
A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inf. Theory*, vol IT-25, pp. 373-380, 1979.

[Gra80]
R.L. Graham and N.J.A. Sloane, "Lower Bounds for Constant Weight Codes," *IEEE Trans. on Inform. Theory*, vol IT-26, no. 1, pp. 31-43, Jan. 1980.

[Gre84]
J.W. Greene and K.J. Supowit, "Simulated Annealing without Rejected Moves," *to appear in ICCD 1984*.

[HemUN]
L.A. Hemachandra and V. K. Wei, "A note on the size of spherical codes," unpublished.

[JohUN]
D.S. Johnson, *in preparation*.

[Joh84]
D.S. Johnson, L. McGeoch, C. Rodriguez, and C. Schevon, "Optimization by Simulated Annealing: an Experimental Evaluation," *Workshop on Statistical Physics in Engineering and Biology*, April 1984.

[Kir83]
S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, 220, 671-680, 13 May 1983.

[Lun83]
M. Lundy and A. Mees, "Convergence of the Annealing Algorithm," unpublished manuscript, Univ. of Cambridge, 1983.

[Mac77]
F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-correcting Codes*, North-Holland, 1977.

[Met53]
N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. of Chemical Physics,.ec* vol. 21, no. 6, June 1953.

[Rom84]
F. Romeo, C. Sechen, and A. Sangiovanni-Vincentelli, "Simulated Annealing Research at Berkeley," *to appear in ICCD 1984*.

[She80]
L. Shepp, "Computerized tomography and nuclear magnetic resonance," *J. Comp. Assisted Tomog.*, vol. 4, no. 1, pp. 94-107, Feb. 1980.

[Slo81]
N.J.A. Sloane, "Tables of sphere packings and spherical codes," *IEEE Trans. on Information Theory*, vol. IT-27, no. 3, pp. 327-338, May 1981.

[SloUNa]
N.J.A. Sloane, "Recent Bounds for Codes, Sphere Packings and Related Problems Obtained by Linear Programming and Other Methods," *unpublished manuscript*.

[SloUNb]
N.J.A. Sloane and J. H. Conway, *private communications*.

[Sua84]
R. Suaya, "Annealing algorithms and VLSI control structures," *to appear in ICCD 1984*.

[Vec83]
M.P. Vecchi and S. Kirkpartick, "Global Wiring by Simulated Annealing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-2, no. 4, pp. 215-222, Oct. 1983.

[Whi84]
S. White, "Concepts of Scale in Simulated Annealing," *to appear in ICCD 1984*.

Figure 0

```
set initial state, x, and temperature, T
do
 {
  do
   {choose y, a perturbation of x
    let ΔE = energy(y) - energy(x)
    if ΔE<0 then x:= y
    else with probability exp(-ΔE/T) x:= y
   }
  until (state has stablized or we've done many iterations)
  decrease temperature
 }
until (a series of stable temperatures is seen)
```

Figure 1

Lower Bounds for Constant Weight Codes		
dimension	distance	Annealing Bound
21	9	22
22	9	23
23	7	17
23	8	24
23	9	24
23	10	39
23	11	39
24	8	28
24	9	24
24	11	57
24	12	60

Figure 2

Size 33 constant weight code for A(24,10,8)

1. 000000000101100011101010
2. 000000100100111001010100
3. 000000110110000100101100
4. 000001000011111000001001
5. 000001010100010110000011
6. 000010001110011010100000
7. 000010101000000001100111
8. 000100001010010000011110
9. 000100110001011000100010
10. 000101010001000001010101
11. 001000000011001110000110
12. 001001101100001000001010
13. 001011000010000011011000
14. 001011011000100100000100
15. 001100101000010111000000
16. 001110000100100000010011
17. 010000000010010101110001
18. 010001110010100000010010
19. 010100001100001100000101
20. 010110010100010001001000
21. 010111001001000010000010
22. 011000001001101000110000
23. 100000101111000000010001
24. 100001100000001110110000
25. 100010001001001101001000
26. 100010010001110010010000
27. 100100100000100010001101
28. 100101000110100101000000
29. 101000010100001001100001
30. 110000011010000011000100
31. 110001000101010000100100
32. 111000000000110100001010
33. 111110100010000000100000

Size 18 constant weight code for A(23,10,7)

1. 000000100100000111100001
2. 000001010110100100001000
3. 000011010000000100010101
4. 000100001010000111000100
5. 000100010001010100000011
6. 000110000000011100101000
7. 001010101010010000000001
8. 001100100000100000010110
9. 010000001100000000001111
10. 010010000011100001000010
11. 010101100101001000000000
12. 011000010001000001110000
13. 100000001101110000100000
14. 100000111000000100100010
15. 100010100001000100011100
16. 101110010100000001000000
17. 110001000000001001100100
18. 111000000000010110000001

Figure 3
Spherical Code Lower Bounds
in Three Dimensions

Angle		Lower bound for M(3,θ)		
cos θ	θ	Lattice	AP	Annealing
~2/3	.841		14	18
13/16	~.62	12	30	31
5/6	~.586	24	34	35
	.524		40	45
~9/10	.45		56	59
13/14	~.38	48	82	83
	π/9		98	99

Figure 4
Size 35 spherical code for M(3 , arccos(5/6))

0:	−0.183291	0.022164	0.982809
1:	0.100781	−0.959698	−0.262341
2:	0.790523	−0.462605	0.401335
3:	−0.810027	−0.387179	−0.440396
4:	0.765679	0.579538	0.279056
5:	−0.269579	0.954104	0.130435
6:	0.302644	−0.846476	0.438047
7:	0.623330	−0.776523	−0.092038
8:	−0.066041	0.897189	−0.436681
9:	−0.208099	0.471576	−0.856919
10:	−0.129810	−0.588486	0.798018
11:	−0.255299	0.666100	0.700809
12:	0.761180	0.103834	0.640175
13:	−0.973947	−0.001453	0.226773
14:	0.874298	−0.322242	−0.362993
15:	0.258973	0.858889	0.441863
16:	−0.615145	0.670210	−0.415228
17:	−0.803612	−0.577832	0.142539
18:	−0.233786	−0.571646	−0.786489
19:	−0.652256	−0.345887	0.674481
20:	0.840418	0.417846	−0.345113
21:	0.291185	0.399403	0.869304
22:	0.620219	0.023453	−0.784078
23:	0.397097	−0.587405	−0.705173
24:	0.995169	0.034284	0.092000
25:	−0.678789	0.259884	0.686809
26:	−0.924050	0.182779	−0.335743
27:	−0.775339	0.603816	0.185083
28:	−0.294027	−0.908184	0.297910
29:	0.378628	0.576793	−0.723845
30:	0.447649	0.881887	−0.147941
31:	0.384192	−0.297955	0.873853
32:	0.069441	−0.068884	−0.995205
33:	−0.555841	−0.007377	−0.831256
34:	−0.466423	−0.836125	−0.288695

NUMBER OF INFORMATION SYMBOLS OF BCH CODES OF LENGTH $2^m + 1$

ALBERTO CACERES
Department of Mathematics, Humacao University College,
CUH Station, Humacao, P. R. 00661

OSCAR MORENO
Department of Mathematics, University of Puerto Rico,
Río Piedras, P. R. 00931

Abstract - This paper offers a solution to the problem of finding the number of information symbols of a binary BCH code of length $2^m + 1$. In the general case we give a method to carry on the counting of the number of information symbols and in the particular case of designed distance $\delta = 2t + 1$ where $2t - 1 < 2^{[m/2]} + 1$ we prove that the dimension is equal to $2^m + 1 - 2mt$. As a consequence we can also give the enumeration of information symbols of Goppa codes of length 2^m which when extended become cyclic.

I. INTRODUCTION

In the present paper we offer a solution to the problem of finding the number of symbols of information of a binary BCH code length $n = 2^m + 1$.

In the general case we suggest an enumeration method by means of counting the cardinality of cyclotomic cosets modulo $2^m + 1$. This is done by the simple technique of cyclic shifting m-digit binary sequences. Unlike the primitive case ($n = 2^m - 1$), the shifting is not all the time cyclic. This is due to the fact that multiplication by 2 modulo $2^m + 1$, when the leading digit is 1, does not correspond exactly to cyclic shifting, as happens in the primitive case [1]. With the help of suitably altered shiftings, complete enumeration is achieved.

In the particular case of designed distance $\delta = 2t + 1$, where $2t - 1 < 2^{[m/2]} + 1$ we obtain a closed formula, in a result similar to the primitive case given as Corollary 8 p., 263 [2]. The shiftings mentioned above are the key factor in proving the theorem.

The theorem also permits the enumeration of symbols of information of Goppa codes of length 2^m which extended become cyclic. This is a contribution to problem 12.1 of Mac Williams and Sloane [2].

II. CYCLOTOMIC COSETS MODULO $2^m + 1$

Enumeration of information symbols of a BCH code of length n is carried on by measuring the size of cyclotomic cosets modulo n. In the primitive case, it is measured by counting the number of different simple cyclic shiftings of the binary representation of coset representatives. When $n = 2^m + 1$ it is necessary to define some sort of corrected shiftings in order to carry out the enumeration.

A shifting modulo $2^m + 1$ or corrected shifting of the binary sequence $a_{m-1}\ a_{m-2} \cdots a_1\ a_0$ is achieved by a simple cyclic shifting $a_{m-2}\ a_{m-3} \cdots a_1\ a_0\ a_{m-1}$ when the leading digit $a_{m-1}=0$. When $a_{m-1}=1$, this is done by a simple cyclic shifting followed by a formal binary addition of the sequence 11 ... 10. This addition corresponds, complementwise, to the substraction of 2 modulo $2^m + 1$.

Theorem (Size of cyclotomic cosets mod $2^m + 1$). Let C_i be the cyclotomic coset of i modulo $2^m + 1$ where i is its smallest representative. Then

i) $|C_1| = 2m$. If $1 \leq k \leq m$ $\quad C_{2^k} = C_1$

ii) $|C_i|$ = number of different shiftings modulo $2^m + 1$ of the m-digit binary expression of i, provided $1 < i < 2m$ and $i \neq 2^k$.

Proof. Part i) is well known since 2m is the multiplicative order of 2 mod $2^m + 1$; so we shall prove part ii).

If $i \neq 2^k$ and $i < 2^m$, let's take its m-digit binary representation

$$a_{m-1}\ a_{m-2} \cdots a_1\ a_0 \qquad (1)$$

where at least two of the digits are 1. We can write

$$i = a_{m-1}\, 2^{m-1} + a_{m-2} 2^{m-2} + \ldots + a_1 2 + a_0 < 2^m$$

a) If $a_{m-1} = 0$ then $i < 2^{m-1}$ and $2i < 2^m < 2^m + 1$ and $i \not\equiv 2i \bmod 2^m + 1$;

also, $2i = a_{m-2}\, 2^{m-1} + a_{m-3}\, 2^{m-2} + \ldots + a_1 2^2 + a_0 2^1 + a_{m-1}$

which corresponds to the sequence $a_{m-2}\ a_{m-3} \cdots a_1\ a_0\ a_{m-1}$

and this is a simple cyclic shifting of (1).

b) If $a_{m-1} = 1$ then $2^{m-1} < i < 2^m$, hence $2^m < 2i < 2^{m+1}$. Moreover $2^m + 1 < 2i < 2^{m+1}$, $2i \equiv 2i - (2^m + 1) \bmod 2^m + 1$ and $2i - (2^m + 1) < 2^m - 1$.

In order to obtain a cyclic shifting expression for $2i - (2^m + 1)$ we write

$$2i - (2^m + 1) = (a_{m-2}2^{m-1} + a_{m-3}2^{m-2} + \ldots + a_0 2) + a_{m-1}2^m - (2^m + 1)$$

$$= (a_{m-2}2^{m-1} + \ldots + a_0 2^1 + a_{m-1}) - a_{m-1} + a_{m-1}2^m - 2^m - 1$$

$$= (a_{m-2}2^{m-1} + \ldots + a_0 2^1 + a_{m-1}) - 2$$

Since $a_{m-1} = 1$.

The expression in parenthesis corresponds to a regular cyclic shifting. Substraction of 2 accomplishes for a shifting modulo $2^m + 1$.

Method to count the number of elements of a cyclotomic coset $C_i \bmod 2^m + 1$ for $1 < i < n$ and i odd.

1. $|C_1| = |C_{2^m}| = 2m$
2. If $1 < i < 2^m$ then obtain the m-digit binary representation of i, say $a_{m-1}\, a_{m-2} \ldots a_1\, a_0$, and perform a shifting modulo $2^m + 1$. It is always possible since at least two digits are 1.
3. The number of different shiftings modulo $2^m + 1$ is the number of elements of C_i.

Example. We calculate $|C_{13}| \bmod 2^6 + 1$. The 6-digit binary representation of 13 is 001101.

001101 — Cyclic shifting
011010 — Cyclic shifting
110100 — Corrected shifting
100111 — Corrected shifting
001101

The last line is equal to the first one, hence $|C_{13}| = 4$.

III. PARTICULAR CASE

Let j be an odd integer such that $1 < j < 2^{[m/2]} + 1$ and let $w_0 = 0\ 0\ \ldots\ 0\ 1_\alpha\ X_{\alpha-1}\ \ldots X_1\ 1_0$ be its m-digit binary representation where subcripts represent the original position of digits and will remain doing so. Since $w_0 < 2^{[m/2]} + 1$, we have $\alpha \leq [m/2] - 1$ and the number of leading zeros in w_0 is $m - \alpha - 1$. Clearly $m - \alpha - 1 \geq m - [\frac{m}{2}] \geq [\frac{m}{2}]$. Let's call w_k the k-th shifting mod $2^m + 1$ of w_0. Now we shall prove the following Lemma.

Lemma. Let j be an odd integer such that $1 < j < 2^{[\frac{m}{2}]} + 1$.

i) If $0 \leq k_1 < k_2 < m$ then $w_{k_1} \neq w_{k_2}$.

ii) For any positive $k < m$, $w_0 < w_k < w_m$, i.e., w_m is the largest of the first $m + 1$ elements of the cyclotomic coset C_j.

Proof. By shifting w_0 for the first time we obtain

$$w_1 = 0\ \ldots\ 0\ 1_\alpha\ X_{\alpha-1}\ \ldots\ X_1\ 1_0\ 0$$

which is a simple shifting since the leading digit is zero. Simple shiftings will occur until we get

$$w_{m-\alpha-1} = 1_\alpha\ X_{\alpha-1}\ \ldots\ X_1\ 1_0\ \underbrace{0\ 0\ \ldots\ 0}_{m-\alpha-1}$$

Clearly all these shiftings are different. By the position of 1_α we get $w_0 < w_k$ for $0 < k \leq m - \alpha - 1$. Next we have a corrected shifting, i.e., move the leading 1_α to the right end and obtain $X_{\alpha-1}\ \ldots\ X_1\ 1_0\ 0\ \ldots\ 0\ 1_\alpha$ and substract 10 to obtain

$$w_{m-\alpha} = X_{\alpha-1}\ \ldots\ 1\ 0_0\ \underbrace{1\ \ldots\ 1}_{m-\alpha-1}\ 1_\alpha$$

which is also different than the previous shiftings and $w_0 < w_{m-\alpha}$.

$$w_{m-\alpha+1} = \begin{cases} X_{\alpha-2} \cdots X_1\ 0_0\ 1_{m-1} \cdots 1_{\alpha+1}\ 1_\alpha\ 0_{\alpha-1} & \text{If } X_{\alpha-1} = 0 \\ X_{\alpha-2} \cdots X_1\ 0_0\ \underbrace{1_{m-1} \cdots 1_{\alpha+1}}_{m-\alpha-1}\ 0_\alpha\ 1_{\alpha-1} & \text{If } X_{\alpha-1} = 1 \end{cases}$$

Next shiftings will move the remaining X's one by one to the right end and, be them 0 or 1, the block of $m-\alpha-1$ 1's will not change until 1_{m-1} reaches leading position and we obtain

$$w_m = 1_{m-1}\ 1 \cdots 1_{\alpha-1}\ Y_\alpha\ Y_{\alpha-1} \cdots Y_1\ 0_0$$

which is also the negative of $w_0 \bmod 2^m + 1$, since $w_m = w_0 2^m \bmod 2^m + 1$. At this point all shiftings have created $m + 1$ different elements all of them larger than w_0. Moreover, since w_m has the largest leading block of 1's it is the largest of all shiftings, and this completes the proof of i) and ii). □

Theorem. For $n = 2^m + 1$ the cyclotomic cosets $C_1, C_3, C_5, \ldots C_i$ are distinct and each contains $2m$ elements provided $i < 2^{[m/2]} + 1$.

Proof. We prove first that each of these cyclotomic cosets contains $2m$ elements. This is a direct consequence of the first part of the Lemma, since the shifting generates $m + 1$ elements in C_j and $|C_j|$ is a divisor of $2m$.

In order to prove that the cyclotomic cosets are different, let's observe that for any $0 \leq k < m$, $w_k 2^m + w_k = w_k(2^m + 1) \equiv 0 \bmod 2^m + 1$, hence $-w_k = w_k 2^m \equiv w_{m+k} \bmod 2^m + 1$, since w_{m+k} is obtained from w_k after m shiftings. Hence $w_k + w_{m+k} = 2^m + 1$. Now, for any $0 < k < m$, the second part of the Lemma tells $w_0 < w_k < w_m < 2^m + 1$ which substracted from $2^m + 1$ gives

$$(2^m + 1) - w_m < (2^m + 1) - w_k < (2^m + 1) - w_0 ,$$

i. e.,

$$w_0 < w_{m+k} < w_m$$

as long as $j < 2^{[m/2]} + 1$. No room is left for any $i < j$ in C_j. Hence $C_i \neq C_j$. □

Note. If m is even and $j = 2^{m/2} + 1$, C_j contains an element $i = 2^{m/2} - 1 < j$. In fact, $j\, 2^{m/2} = (2^{m/2} + 1)\, 2^{m/2} = (2^{m/2}-1) + (2^m + 1) \equiv i \bmod 2^m + 1$.

Corollary. Let C be a binary BCH code of length $n = 2^m + 1$ and designed distance $\delta = 2t + 1$ where $2t - 1 < 2^{\lceil m/2 \rceil} + 1$.

Then $\dim C = 2^m + 1 - 2mt$.

ACKNOWLEDGMENT

This work was jointly supported by the NSF-Resource Center for Science and Engineering, UPR, and by the Humacao University College, UPR.

REFERENCES

[1] Berlekamp E. R., "Algebraic Coding Theory", McGraw Hill, 1968.

[2] Mac Williams F. J. and Sloane N. J. A., "The Theory of Error-Correcting Codes", North Holland Pub. Co., Third Printing, 1981.

TRANSFORM APPROACH TO GOPPA CODES

MANSOUR LOELOEIAN and JEAN CONAN
Ecole Polytechnique de Montreal
Department of Electrical Engineering
Montreal, Canada

ABSTRACT

Using the finite Fourier transform of codewords over the finite field $GF(q^m)$, some of the properties of Goppa codes are described. A new lower bound on the minimum distance of these codes is presented. Furthermore, using these results, it is shown that a recently found [56,16,20] binary Goppa code is in fact a punctured reversible cyclic code and a general class of such codes will be outlined. A large class of codes, known as Goppa codes, were introduced by Goppa [1,2]. Goppa codes are exceptionally good in the sense that they asymptotically meet the Varshamov-Gilbert bound and also include the class of BCH and Srivastava codes. Berlekamp and Moreno [3], Tzeng and Zimmermann [4] have recently shown that certain multiple-error correcting q-ary Goppa codes becomes cyclic when extended by an overall parity check and thus revealed another close relationship between Goppa codes and cyclic codes.

In this presentation, a Fourier type of transform is proposed which is useful in deriving properties of Goppa codes. It is found that this transformation includes the so called Mattson-Solomon polynomials in the special case for which n divides q^m-1, where n is the block length of the code. Using these results, a new lower bound for the minimum distance of Goppa codes is derived. Also it is shown that a subclass of Goppa codes recently considered by the authors [6] can in fact be constructed from punctured reversible cyclic codes. Finally a running example is discussed which shows partial applicability of these results.

REFERENCES

[1] V.D. Goppa, "A new Class of Linear Error Correcting Codes", Probl. Peredach. Inform., Vol. 6, pp. 24-30, Sept. 1970.

[2] V.D. Goppa, "Rational Representation of Codes and (L,g) Codes", Probl. Peredach. Inform., Vol. 7, pp. 41-49, Sept. 1971.

[3] E.R. Berlekamp and O. Moreno, "Extended Double-Error Correcting Binary Goppa Codes are Cyclic", IEEE Trans. Inform. Theory, Vol. IT-19, pp. 817-818, Nov. 1973.

[4] K.K. Tzeng and K. Zimmermann, "On Extending Goppa Codes to Cyclic Codes", IEEE Trans. Inform. Theory, Vol. IT-21, pp. 712-716, Nov. 1975.

[5] F.J. MacWilliams and N.J.A. Sloane, "The Theory of Error Correcting Codes", New York and Amsterdam, North Holland 1977.

[6] M. Loeloeian and J. Conan, "A [55, 16, 19] binary Goppa code", IEEE Trans. Inform. Theory, Vol. IT-29, Sept. 1984.

TREE CODING OF SPEECH WITH BLOCK-ADAPTIVE FILTERS

William A. Pearlman and Mark Johnson
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

ABSTRACT

In tree coding of speech, the output of the coding filter with i.i.d. Gaussian sequences as input is the reproduction or codeword sequence process. A recent theorem [1] has established the optimal power spectrum of the reproduction process and hence the magnitude of the frequency response of the optimal coding filter $H(\omega)$ for tree encoding of stationary Gaussian sources with the squared error criterion for all rates. Some researchers [4,5,6] have used another coding filter ($G(\omega)$) in order to have the power spectrum codeword sequence process match that of the source. Here, low order approximations to the $H(\omega)$ and $G(\omega)$ filters are used to tree-encode speechlike data and actual speech at rates of 1.0 and 0.5 bits per source symbol. For actual speech, the coding filters are adapted in a blockwise fashion based on a power spectrum estimate of the current source block. The performances of these filter approximations are compared in terms of signal-to-noise ratios and informal listening tests as a function of increasing search intensity in the tree encoding algorithm. It is shown that for small search intensities the G-filter coding is somewhat better, but as the search increases the optimally-derived H-filter coding becomes superior. It is also shown that a fourth order filter gives sufficiently good results. The highest signal-to-noise ratio, obtained with H-filter coding exceeds by 2 dB a recently reported comparable G-filter block-adaptive tree coding result [6].

INTRODUCTION

The objective of this paper is to compare the mean squared-error performances of tree codes using rate-distortion-derived and DPCM-type coding filters for speech and speech-like signal sources. The impetus for such a study stems from the recent theorem of Mazor and Pearlman [1] that in optimal tree coding of a discrete-time stationary Gaussian source with a power spectrum $S_x(\omega)$ satisfying the Lip 1 condition the optimal reproduction process is Gaussian with power spectrum $S_y(\omega)=\max\{S_x(\omega)-\theta,0\}$, where θ is the parameter specifying a particular point of the rate-distortion function. This solution for $S_y(\omega)$ holds for all rates and subsumes the proven result of Berger [2] valid only for autoregressive sources above the so-called critical rate, i.e., when $S_x(\omega) \geq \theta$ for all $\omega \in [-\pi, \pi]$. Dick [3] and Berger [2] conjectured the above solution for the optimal $S_y(\omega)$ for all rates, but did not prove the theorem. A Gaussian reproduction process with the power spectrum $S_y(\omega)$ can be generated by passing a sequence of N(0,1) (normal, mean 0 and variance 1) and independent random variables through a coloring filter of frequency response magnitude $|H(\omega)| = \sqrt{\max\{S_x(\omega)-\theta,0\}}$. In a tree code, the independent N(0,1) sequences are attached to the branches of a tree with the desired code rate. Another coloring filter motivated by DPCM and the fake process approach of Linde and Gray [4], has frequency magnitude

$|G(\omega)| = \sqrt{S_x(\omega)}$. For high rates Θ is small and the two filters are close in response, but for rates below the critical rate when Θ is comparable to $S_x(\omega)$ over some range of ω, the filter responses exhibit significant differences. Stewart, Gray, and Linde [5] and Fehn and Noll [6] have encoded speech using tree and trellis codebooks with the filter $G(\omega)$. Wilson and Husain [7] have tree encoded speech through the filter $H(\omega)$ for a frequency-weighted, squared error criterion, despite the absence of a corroborating theorem. Their codes, however, were multi-path DPCM, where the filter inputs are binary valued instead of being normal variates.

One obstacle to encoding below the critical rate is that since $\Theta > S_x(\omega)$ for some values of ω, the desired $H(\omega)$ contains a null zone in its frequency response. Then a finite order digital filter cannot match $H(\omega)$, but merely approximate it. One issue is how low can the filter order be and still give a sufficiently good coding result. This question is addressed here, in part, by encoding an autoregressive source and actual speech using all-pole and pole-zero approximations of various orders to the desired filters. The main objective, however, was to compare results of parallel simulations with the same order approximations to the optimal $H(\omega)$ and the DPCM-type $G(\omega)$. For the actual speech simulations, the filter was adapted for successive blocks of 256 speech samples (lasting 32 milliseconds) through a power spectrum estimate of the code. In order to decode block-adapted speech, the updated filter coefficients must be transmitted every block along with the digitized speech.

TREE CODING

In a tree code, every codeword or reproduction sequence is stored on the branches of a tree as a concatenation of branch symbols along a path through the tree. For a tree with α branches emanating from each node and β symbols on each branch, the code rate is $R = \beta^{-1} \log_2 \alpha$ bits per source symbol. For the optimal random codes treated here, the reproduction sequences in the tree are selected at random from an optimal probability distribution found by solving the rate-distortion function for the given Gaussian source and squared-error distortion measure. Mazor and Pearlman [1] have proved a tree coding theorem valid for stationary Gaussian sources with power spectrum $S_x(\omega)$ obeying the Lip 1 condition ($|S_x(\omega_2) - S_x(\omega_1)| \leq K\,|\omega_2 - \omega_1|$ for some constant K and all ω_1, $\omega_2 \in [-\Pi, \Pi]$) and the squared error criterion. They also proved that the optimal reproduction process is Gaussian with power spectrum $S_y(\omega) = \max [S_x(\omega) - \Theta, 0]$. This theorem and spectral result confirm the conjecture of Dick [3] and Berger [2] and subsumes the theorem of Berger [2] valid only for Gaussian autoregressive sources above the so-called critical rate R_c or below the critical distortion $D_c = \inf_{\omega} S_x(\omega)$.

In order to construct a tree code with sequences of proper statistics, the branches of a tree are populated with independent, identically distributed (i.i.d.) normal variates of unit variance (N(0,1), where the number β of symbols per branch is set to give the desired code rate. When any sequence of symbols along a path through this "white" tree is input to a causal filter, the resulting sequence can be deposited on an identical rate tree with the same path map as that of the input. Therefore, a "colored" tree results at the output, which can then be searched for the

codeword sequence that best matches the given source sequence according to the chosen distortion measure. The path map of this sequence is then released to the destination, at which is stored the same white tree and coloring filter for decoding. The coloring filters used here are the optimal $|H(\omega)| = \sqrt{\max[S_x(\omega) - \theta, 0]}$ and suboptimal $|G(\omega)| = \sqrt{S_x(\omega)}$. For the purpose of saving computation, a suboptimal, selective search procedure known as the M-algorithm was chosen in all cases. The parameter M refers to number of paths retained at each tree level.

CODING OF AN AUTOREGRESSIVE SOURCE

Encoding simulations using $H(\omega)$ and $G(\omega)$ were first carried out for a Gaussian autoregressive (AR) source (1.5, -0.79, 0.07) which models the long-term statistics of speech. The code rate was 1.0 bit per source symbol, which is well below the critical rate of 1.748 for this source. A calculation of the rate-distortion function at rate R=1 shows that the maximum theoretical signal-to-noise ratio is 13.99 dB. (Signal-to-noise ratio in dB = $10 \log_{10} \frac{\text{signal variance}}{\text{noise variance}}$.) The i.i.d. N(0,1) variates populated a binary (α=2) branching tree with one symbol to each branch (β=1). The same source sequence was encoded regardless of whether $H(\omega)$ or $G(\omega)$ followed the white tree to yield the code. The filter $G(\omega)$ in this case is exact, as it duplicates the source generation process given by the AR coefficients. A 3-pole, 4-zero approximation to the optimal $|H(\omega)|$ is synthesized with Deczky's algorithm [8], following Mazor and Pearlman's observation that the number of poles emulate the order of the AR source [1].

A total of ten 250-sample blocks were individually encoded using each filter and search values M of 2, 4, 8, 16, and 32. The results are shown in Table 1. At the low values of M=2 and M=4, average signal-to-noise ratio (SNR) results with suboptimal G are somewhat better than with H, and considerably below the theoretical maximum SNR of 13.99 dB. But at M=8 and above the SNR's obtained with H surpass those of G by an amount increasing with M. At M=32 the H-filter SNR is only 1.3 below the theoretical maximum. This trend is in agreement with the off-stated fact that the promise of optimal codes can be realized only for high search effort.

BLOCK-ADAPTIVE CODING OF SPEECH

In order to encode actual speech at rates of 1.0 bit per sample and below with fairly intelligible reconstructions, it is necessary to adapt the code to the changing speech modes. As corroboration of this necessity, the work of Anderson and Bodie in their pioneering paper [9], where they encoded speech non-adaptively with rate-distortion derived and DPCM code trees at a rate of 1.0 bit per source symbol, produced barely intelligible reconstructed speech. Two possible types of adaptation to consider are sequential and block adaptation. In sequential adaptation, the coding filter is adjusted every sample according to the statistics of the past reproduction symbols. In such a scheme, where the reproduction symbols are available at the receiver, no extra information needs to be transmitted for the purposes of code adaptation. One could also attempt the same kind of adaptation called feedback adaptation using reproduction sequences on a blockwise basis. Adaptation can also be enacted blockwise according to the changing statistics of the source sequence, which requires the transmission of the updated filter coefficients every block for decoding. Feedback adaptation is very susceptible to a relatively small amount of channel

noise. Moreover, noiseless feedback sequential adaptation was investigated for comparison purposes and gave noticeably inferior results to non-feedback block adaptation for the same amount of total rate. For this reason, only non-feedback block adaptation and the results are reported here.

The speech data are two sentences low-pass filtered to 3.2 kHz and sampled at a rate of 8 kHz: (1) "The pipe began to rust while new", (female speaker) and (2) "Oak is strong and also gives shade", (male speaker). Each sentence is segmented into 60 blocks of 256 samples per block of 32 milliseconds. For each segment, the power spectrum is estimated and the approximation to the magnitude of the frequency response of the filter $G(\omega)$ or $H(\omega)$ is synthesized by Deczky's algorithm [8]. Examples of three pole approximations to the desired filters $G(\omega)$ and $H(\omega)$ at the rates of 1.0 and 0.5 bits per source symbol (bits/ss) are shown in Figure 1. Note the difference in cutoff frequency and peak approximation of the rate-independent $G(\omega)$ from the optimal $H(\omega)$ at the rate of 0.5 bits/ss.

The first set of comparisons uses these 3-pole approximations to the desired $G(\omega)$ and $H(\omega)$ filters for encoding each segment at a code rate (neglecting overhead rate) of 1.0 bit per sample or 8,000 bps. The results are calculated in terms of overall SNR and average segmental SNR (SEGSNR) in dB and shown in Table 2. Again, as M increases SNR and SEGSNR of the optimally derived filter eventually surpass the corresponding figures of the suboptimal one. The trend and superiority of the filter H are more noticeable perceptually than the dB differences indicate. One also notices very similar SNR and SEGSNR figures when comparing the same cases for the sentences. At M=32, and R=1.0 bit/ss, for example, one sentence has a SEGSNR of 12.50 for the filter G versus 12.61 for H, while the other sentence's corresponding figures are 12.56 and 12.71. The robustness of tree-searched codes is important for applications.

Suspecting that better perceptual and signal-to-noise ratio results could be obtained with closer filter approximations, simulations with two higher order filter approximations were carried out for Sentence 1 at 1.0 bit per sample. The designations of poles and zeros were 4,4 and 5,5 respectively. The results are given in Table 3. Both filter approximations produced noticeable perceptual improvements over the 3-pole case, as indicated by increases in SEGSNR. The 1 dB gain in SEGSNR for the optimal H-filter was twice that of the G-filter at M=32. Listening to the reconstructed speech showed the corresponding perceptual effect. Note that only slight further improvements resulted from raising the poles and zeros from 4 to 5. In an attempt to ascertain whether the additional complexity associated with the synthesis of zeros is warranted, all-pole filters of orders 2 through 6 were used for the sentence at rate 1.0 bit per sample and search M=16 with the results shown in Table 4. The SEGSNR and SNR do improve significantly in raising the order from 2 to 4, but then level off. As the figures for the simple 4-pole and 4 pole, 4-zero structures differ only by 0.1 dB, these results suggest that a simple 4 pole structure is adequate and that the extra computation needed for more elaborate filter structures is better spent for increasing the search intensity.

One final remark should be made about the overhead information of the filter coefficients, which must be transmitted every block of 256 samples or 32 milliseconds. Even with the conservative assumption of six bits per coefficient, a 4-pole, 4-zero filter would require an overhead of 48 bits or an extra information rate of 0.19 bits/s.s. or 1,500 bps for straight

PCM encoding. This overhead rate, though relatively small, can certainly be reduced further through assignment of fewer bits to less important coefficients and a more efficient encoding scheme, such as DPCM.

SUMMARY AND CONCLUSIONS

Random tree codes generated through an optimally derived coding filter have been shown to be superior to those generated through a source-matched DPCM-type filter as the search intensity increases when encoding a speechlike AR source non-adaptively and actual speech using blockwise adaptation. In all but one case, the crossover point for superiority of the optimally derived filter occurs at M=8, which means that 16 branches are searched for every source symbol. It was also determined that a 4-pole, 4-zero filter approximation gave very good signal-to-noise ratio results in block-adaptive coding of actual speech. Even a single all-pole filter of order four gave nearly as good results. In listening tests, the speech was certainly intelligible and sounded natural, but some coding noise was also noticeable.

Comparison to results of other investigators with tree-coded speech is complicated by the use of different speech sentences and different error criteria. Wilson and Husain [7] reported their block-adapted speech results for multipath DPCM for M=4 in terms of frequency-weighted signal-to-noise ratios, which have a different meaning than ours. Fehn and Noll [6] ascertained improvements of 2 to 4 dB in SEGSNR when tree search is used in conjunction with single-path adaptive DPCM. Their best SEGSNR result of 11.3 dB at rate 1.0 is more than 2 dB below our best result of 13.5 dB obtained with the 4-pole, 4-zero block-adaptive H-filter. Considering the closeness of the SNR and SEGSNR values in our simulations with two different sentences with different gender speakers, a reasonable conjecture is that the 2 dB difference is attributable mostly to the coding methods.

REFERENCES

1. B. Mazor and W. A. Pearlman, "An Optimal Approach for Time-Domain Tree Coding of Stationary Gaussian Sources", Proceedings of the Twentieth Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, October 1982, pp. 689-696.

2. T. Berger, Rate Distortion Theory, Prentice-Hall 1971.

3. R. J. Dick, Tree Coding for Gaussian Sources, Ph.D. dissertation, Cornell University, Electrical Engineering Dept., 1973.

4. Y. Linde and R. M. Gray, "A Fake Process Approach to Data Compression", IEEE Trans. on Communications, Vol. COM-26, pp. 840-847, June 1978.

5. L. C. Stewart, R. M. Gray, and Y. Linde, "The Design of Trellis Waveform Coders", IEEE Trans. on Communications, Vol. COM-30, pp. 702-709, April 1982.

6. H. G. Fehn and P. Noll, "Multipath Search Coding of Stationary Signals with Applications to Speech", IEEE Trans. on Communications, Vol. COM-30, pp. 687-701, April 1982.

7. S. G. Wilson and S. Husain, "Adaptive Tree Encoding of Speech at 8000 Bits/s with a Frequency-Weighted Error Criterion", IEEE Trans. on Communications, Vol. COM-27, pp. 165-170, Jan. 1979.

8. A. G. Deczky, "Synthesis of Recursive Digital Filters Using the Minimum P-Error Criterion", IEEE Trans. on Audio Electroacoustics, AU-20, pp. 257-263, Oct. 1972.

9. J. B. Anderson and J. B. Bodie, "Tree Encoding of Speech", IEEE Trans. on Information Theory, Vol. IT-21, pp. 379-387, July 1975.

ACKNOWLEDGEMENT

This research was sponsored by the Air Force Office of Scientific Research under Grant No. AFOSR-81-0188.

Table 1

Tree Encoding Simulations of AR Source (1.5, -0.79, 0.07)*

Search Parameter M	SNR** G(ω)	H(ω)
2	7.73	5.55
4	9.95	8.96
8	10.94	10.98
16	11.80	12.13
32	12.39	12.70

*α=2, β=1, R=1.0 bit/s.s, SNR_{max} = 13.99 dB

**Ratio of signal variance to mean-squared error for 10 blocks of 250 samples in dB.

Table 2

SNR and SEGSNR for Speech with 3-Pole Block-adaptive Filters*

M	G-Filter SNR	G-Filter SEGSNR	H-Filter SNR	H-Filter SEGSNR
	(a) Sentence 1, R=1.0 bit/s.s.			
4	11.25	10.68	11.24	10.75
8	12.20	11.56	12.39	11.56
16	12.80	12.08	13.01	11.94
32	13.24	12.50	13.49	12.61
	(b) Sentence 2, R=1 bit/s.s.			
4	11.21	10.69	11.16	10.62
8	12.10	11.49	12.28	11.53
16	12.75	12.00	12.96	12.10
32	13.32	12.56	13.43	12.71
	(c) Sentence 1, R=0.5 bit/s.s.			
4	7.93	7.80	8.08	7.84
8	8.95	8.64	9.02	8.38
16	9.25	8.94	9.57	8.62
32	9.66	9.31	9.91	9.44

*60 blocks of 256 samples per block. (α=2, β=1 tree)

Table 3

SNR and SEGSNR vs. Search Intensity for Sentence 1 at R=1.0 bit/ss with Higher Order Filters*

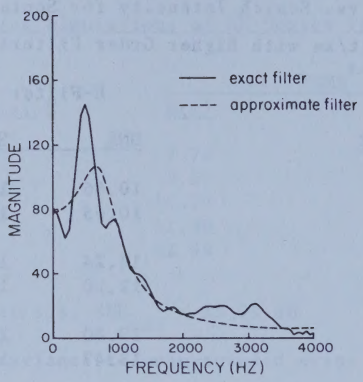
	G-Filter		H-Filter	
M	SNR	SEGSNR	SNR	SEGSNR
4	9.80	9.12	10.66	10.13
	10.16	9.72	10.65	10.07
8	12.18	11.69	12.24	11.64
	12.13	11.70	12.20	11.61
16	13.19	12.53	13.40	12.69
	13.22	12.72	13.47	12.79
32	13.59	13.04	14.00	13.50
	13.69	13.15	14.06	13.62

*Top entry for 4-pole, 4-zero filter, bottom for 5-pole, 5-zero filter 60 blocks of 256 Samples per block.

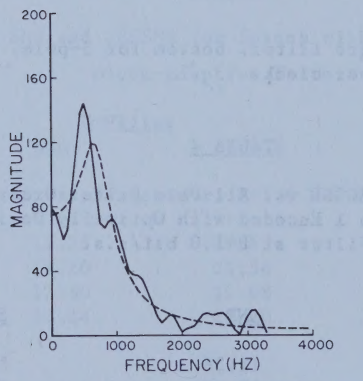
Table 4

SNR and SEGSNR vs. All-Pole Filter Order for Sentence 1 Encoded with Optimally Derived H-Filter at R=1.0 bit/s.s.

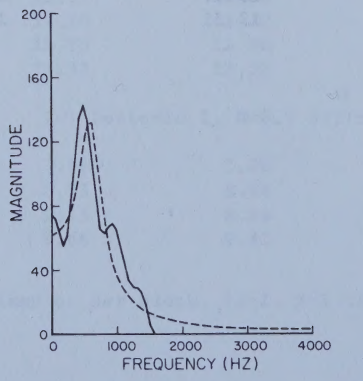
Order	SNR	SEGSNR
2	12.36	11.12
3	13.10	12.04
4	13.40	12.69
5	13.47	12.80
6	13.51	12.82



(a) Suboptimal Coloring Filter ($G(\omega)$)



(b) Optimal Coloring Filter ($H(\omega)$) at R = 1.0 bit/s.s.



(c) Optimal Coloring Filter ($H(\omega)$) at R = 0.5 bits/s.s.

Figure I. Three-pole Filter Approximations

HYBRID TRANSFORM/DPCM CODING OF IMAGES USING PYRAMID VECTOR QUANTIZATION

Thomas R. Fischer and Huey-Chen Tseng
Department of Electrical Engineering
Texas A&M University
College Station, TX 77843

ABSTRACT

Hybrid transform/DPCM coding of images is generalized to allow pyramid vector quantization (PVQ) of the transform coefficients. An asymptotic mean-squared error performance expression is derived for the PVQ, and used to determine the optimum rate assignment for encoding the various transform coefficients. Coding simulations for four images demonstrate a two to three decibel improvement in signal-to-noise ratio for the vector quantization approach. For low rates, this improvement is quite noticeable.

I. INTRODUCTION

Transform coding [1] and hybrid transform/differential pulse code modulation (DPCM) coding [2] are popular techniques for the data compression for images. The transform coding approach uses a linear transformation (typically, the discrete cosine transform [3]) of an $n \times n$ block of image pixel intensities, to approximate the Karhunun-Loeve transform of the data. The individual transform coefficients are then quantized, with a nonuniform bit assignment [4] depending on the energy (variance) of each transform coefficient. The hybrid coding approach utilizes a one-dimensional transform (say, in the horizontal direction) and DPCM encoding of the transform coefficients (in the vertical direction). Again, bits are allocated in a nonuniform fashion depending on the importance of the various coefficients. Both the transform and hybrid transform/DPCM coding techniques can be improved by using vector rather than scalar quantization. Such a conclusion follows readily from the inherent superiority of vector quantization (VQ) [5].

There are several approaches that can be used in applying VQ to image coding. Lattice quantizers [6] have been successfully employed by Sayood et al. [7] in transform coding. Alternatively, the Linde, Buzo, and Gray (LBG) [8] clustering algorithm has been used [9,10] in image compression. These latter approaches have not considered transform or hybrid transform coding, presumably because some transform coefficients must be coded at large rates, and vector quantization with large rates and dimension is not presently feasibly with LBG based designs.

Goodness-of-fit tests indicate [11] that with the discrete cosine transform of images, many of the transform coefficients may be adequately modelled as Laplacian. Motivated by this result, we extend the hybrid transform/DPCM coding technique to the VQ case by utilizing the pyramid vector quantizer (PVQ) [12]. The PVQ was designed based on the implicit geometry of an independent and identically distributed

(i.i.d.) Laplacian source. This geometric approach has been shown to be asymptotically optimum in a rate–distortion sense, and gives rise to the easily implemented. lattice–based PVQ.

The paper is organized as follows. Section II presents a brief overview of the PVQ and some asymptotic bounds on performance. Section III extends the hybrid transform coding method to the vector case. Finally, in Section IV numerical results are presented for several images.

II. THE PYRAMID VECTOR QUANTIZER

The pyramid vector quantizer was motivated by characterizing the spatial properties of a sequence of i.i.d. Laplacian random variables. Let vector $\mathbf{x} = (x_1, x_2, \ldots, x_L)$ have i.i.d. Laplacian components, x_i, $i = 1, \ldots, L$, with zero mean and variance $\frac{2}{\lambda^2}$. The joint density for $\mathbf{x}$ is then

$$f_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{L} \frac{\lambda}{2} e^{-\lambda|x_i|} \tag{1}$$

and the differential entropy per degree of freedom is

$$h = \frac{-1}{L} \int \cdots \int f_{\mathbf{X}}(\mathbf{x}) \log f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \log \frac{2e}{\lambda}. \tag{2}$$

Using the law of large numbers it is easy to show that

$$\frac{1}{L} \log f_{\mathbf{X}}(\mathbf{x}) \xrightarrow{\text{a.s.}} -h \text{ as } L \to \infty. \tag{3}$$

That is, for large dimension, only a particular set of vectors $\mathbf{x}$ tend to occur.

Define the contour of constant probability density function (pdf) as

$$\tilde{S}(L, C) = \{\mathbf{x} : \frac{\log f_{\mathbf{X}}(\mathbf{x})}{L} = C\} \tag{4}$$

Substituting (1) into (4) and letting $C = -h$, the contour of constant pdf important for source coding may alternatively be described as

$$S(L, \frac{L}{\lambda}) = \{\mathbf{x} : \sum_{i=1}^{L} |x_i| = \frac{L}{\lambda}\}. \tag{5}$$

Although (3) indicates convergence in a logarithmic sense, for source coding usually a difference distortion measure is used. Let $d_r(\mathbf{x}, \hat{\mathbf{x}})$ be the distortion measure

$$d_r(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_r^r = \sum_{i=1}^{L} |x_i - \hat{x}_i|^r \tag{6}$$

for fixed $r \geq 1$. Then it is readily shown [13] that for arbitrary $\delta > 0$ and $\Delta > 0$, there exists a positive integer L_o, such that for $L > L_o$ there exists a vector $\hat{\mathbf{x}} \in S(L, \frac{L}{\lambda})$ satisfying

$$\frac{1}{L} d_r(\mathbf{x}, \hat{\mathbf{x}}) < \delta,$$

for all vectors $\mathbf{x}$, except on a set of total probability Δ. That is, there exists a vector $\hat{\mathbf{x}} \in S(L, \frac{L}{\lambda})$ such that

$$\frac{1}{L} d_r(\mathbf{x}, \hat{\mathbf{x}}) \longrightarrow 0 \text{ as } L \rightarrow \infty. \tag{7}$$

Vector $\hat{\mathbf{x}}$ is easily computed using the Projection Theorem as

$$\hat{\mathbf{x}} = \mathbf{x} - \left[(\mathbf{x}, \frac{\mathbf{s}}{\|\mathbf{s}\|_2}) - \frac{L\|\mathbf{s}\|_2}{\lambda \|\mathbf{s}\|_1} \right] \frac{\mathbf{s}}{\|\mathbf{s}\|_2} \tag{8}$$

where $(\cdot, \cdot)$ is the usual inner product and $\mathbf{s} = \mathbf{s}(\mathbf{x})$ is given by

$$s_i = \begin{cases} 1 & \text{if } x_i > 0 \\ 0 & \text{if } x_i = 0 \\ -1 & \text{if } x_i < 0 \end{cases}, \text{ for } \|\mathbf{x}\|_1 \geq \frac{L}{\lambda},$$

or

$$s_i = \begin{cases} 1 & \text{if } x_i \geq 0 \\ & \\ -1 & \text{if } x_i < 0 \end{cases} \quad \text{for } \|\mathbf{x}\|_1 < \frac{L}{\lambda}.$$

As a consequence of (7), optimum coding of a memoryless Laplacian source can be accomplished by a) for each $\mathbf{x}$, find the closest (in the sense of (6)) vector $\hat{\mathbf{x}} \in S(L, \frac{L}{\lambda})$; b) optimally encode $\hat{\mathbf{x}}$. The source coding problem then devolves to constructing an optimum code word assignment for points on the pyramid.

The pyramid vector quantizer was developed by selecting all cubic lattice points on a pyramid as code words (or scaled versions of code words). That is, for integer $K \geq 1$, let $N(L, K)$ be defined as

$$N(L, K) = \left(\begin{array}{c} \text{The number of vectors } \mathbf{u} \text{ satisfying} \\ \mathbf{u} \in S(L, K) \text{ and } u_i = \text{ integer, all } i = 1, \cdots, L \end{array} \right).$$

For given values of L and K, $N(L, K)$ can be easily computed [12].

For specified values of L and average rate per dimension, R, the largest value of K should be selected such that

$$N(L, K) \leq 2^{RL}$$

The basic PVQ encoding algorithm is then (i) given $\mathbf{x}$, find $\hat{\mathbf{x}} \in S(L, \frac{L}{\lambda})$ that is closest to $\mathbf{x}$ (in the sense of (6)); (ii) scale $\hat{\mathbf{x}}$ to pyramid $S(L, K)$ according to $\tilde{\mathbf{x}} = K\hat{\mathbf{x}}/\|\hat{\mathbf{x}}\|_1$; (iii) quantize $\tilde{\mathbf{x}}$ to $\tilde{\mathbf{y}}$ (this can be done essentially by rounding each vector component and then adjusting to make sure $\tilde{\mathbf{y}} \in S(L, K)$); (iv) scale $\tilde{\mathbf{y}}$ to form the PVQ output $\mathbf{y} = \alpha \tilde{\mathbf{y}} \cdot \frac{L}{\lambda K}$. (Parameter α varies with the rate and is adjusted to scale the codewords slightly inside the pyramid).

The asymptotic (in dimension) performance of the PVQ can be evaluated by recognizing that a cubic lattice based quantizer is only capable of a normalized (per quantization cell) mean–square error (MSE) distortion of $\frac{1}{12}$, which is the normalized MSE distortion for an optimum scalar uniform quantizer. Interestingly, other lattices (such as A_n and D_n) have this same asymptotic property [6]. Formation of vector $\hat{\mathbf{x}} \in S(L, L)$ in the PVQ encoding requires that

$$\frac{1}{L} \log f_{\mathbf{x}}(\hat{\mathbf{x}}) = -h.$$

Thus, equating the entropy of a uniform random variable (on interval $(-\frac{\Delta}{2}, \frac{\Delta}{2})$) to that of a Laplacian, provides

$$\log \Delta = \log \frac{2e}{\lambda},$$

so that the variance of the uniform random variable, σ_u^2, is

$$\sigma_u^2 = \frac{\Delta^2}{12} = \frac{1}{12}(\frac{2e}{\lambda})^2.$$

Since the optimum uniform scalar quantizer provides a distortion

$$\sigma_u^2 2^{-2R}$$

where R is the rate (corresponding to an integral number of quantization levels), the asymptotic distortion expected from the PVQ (normalized to $\lambda = \sqrt{2}$, a unit variance source) is

$$MSE = \frac{e^2}{6} 2^{-2R}. \tag{9}$$

Note that this is well above the Shannon lower bound for a Laplacian source,

$$MSE_{\text{SLB}} = \frac{e}{\pi} 2^{-2R}.$$

Interestingly, the thresholding operation in the general PVQ encoding algorithm (see [12]) is nonlinear, and improves the performance below that of (8) for low rates ($R = 1$). Without this thresholding operation, the PVQ performance is bounded by (8).

As depicted in Figure 1, the MSE for the PVQ can be approximated as

$$\|\mathbf{x} - \mathbf{y}\|_2^2 \simeq \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \|\hat{\mathbf{x}} - \mathbf{y}\|_2^2. \tag{12}$$

The approximation is exact for all $\mathbf{x}$, except those few that occur "above" a corner of the pyramid. As the vector dimension increases, such vectors occur with vanishing total probability.

Although asymptotic in dimension the first term in (12), namely

$$\frac{1}{L} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2, \tag{13}$$

is negligible, for moderate dimension, this distortion may be important. In fact, for a rate of two bits per dimension, (13) is significant even if $L = 100$. This suggests that the PVQ performance can be improved by using code words on concentric pyramids as the basis for the quantizer. More specifically, let radial parameter r be defined as

$$r = \sum_{i=1}^{L} |x_i|, \tag{14}$$

with mean and variance

$$\begin{aligned} E[r] &= \frac{L}{\lambda}, \\ \mathrm{Var}[r] &= \frac{L}{\lambda^2}. \end{aligned} \tag{15}$$

Since r is the sum of i.i.d. random variables, for large L, $r/\sqrt{L}$ has approximately a normal distribution.

For a fixed L, a product code [14,15] may be defined as having code words $\hat{r}_j \hat{\mathbf{a}}_i$, $j = 1, \ldots, N_1$, $i = 1, \ldots, N_2$. Here the $\hat{r}_j$ correspond to scalar quantized versions of r, and the $\hat{\mathbf{a}}_i$ are points on a normalized pyramid. The number of code words is constrained as

$$N_1 N_2 \leq 2^{RL},$$

and encoding is accomplished by separately scalar quantizing r as $\hat{r}$, and pyramid vector quantizing $\mathbf{x}$. Since $\hat{r}$ indexes the pyramid closest to $\mathbf{x}$, step (i) in the PVQ encoding algorithm is modified to finding $\hat{\mathbf{x}}$ as the closest vector to $\mathbf{x}$ on $S(L, \hat{r})$.

The basic product code PVQ design problem is to select N_1 and N_2 (that is, to allocate a total rate of RL to the scalar and pyramid quantizers), as well as select the scalar quantizer outputs, to minimize the average MSE in (12). Using (8), the first term in (12) can be approximated as

$$\begin{aligned} E\left[\frac{1}{L}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2\right] &= E\left[\frac{1}{L}\left\|\left[(x, \frac{\mathbf{s}}{\|\mathbf{s}\|_2}) - \frac{\|\mathbf{s}\|_2}{\|\mathbf{s}\|_1}\frac{L}{\lambda}\right]\frac{\mathbf{s}}{\|\mathbf{s}\|_2}\right\|_2^2\right] \\ &\simeq E\left[\frac{1}{L}\left|\frac{\|\mathbf{x}\|_1}{\sqrt{L}} - \frac{\sqrt{L}}{\lambda}\right|^2\right] = \frac{1}{L^2}\mathrm{Var}[r]. \end{aligned} \tag{16}$$

Since the distribution of $\|\mathbf{x}\|_1/\sqrt{L}$ approaches a normal distribution for large L, the scalar quantizer for r should be the Max [16] quantizer. The resulting distortion was modeled by Huang and Schultheiss [17] as

$$D(R) = \sigma^2 K(R) 2^{-2R}, \tag{17}$$

where σ^2 is the variance and $K(R)$ is a nondecreasing function of R with $K(0) = 1$. Combining (15), (16), and (17) implies

$$E\left[\frac{1}{L}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2\right] \simeq \frac{1}{L}\left(\frac{1}{\lambda^2}\right) K(R) 2^{-2R}.$$

Let R_r be the rate assigned to the scalar quantizer and R_p the rate assigned to the pyramid quantizer. Using (9) as an approximation for the second term in (12), the rate should be allocated to minimize

$$\frac{e^2}{6}2^{-2R_p} + \frac{1}{L}(\frac{1}{\lambda^2})K(R)2^{-2R_r}$$

subject to the constraint

$$R_r + LR_p = LR.$$

Using the method of Goodman [18], rates R_r and R_p can be easily computed.

III. HYBRID TRANSFORM CODING

The general image coding scheme is depected in Figure 2. The $N \times N$ image is broken up into M vertical stripes of width n (so that $nM = N$). A 1–dimensional (n–point) discrete cosine transform (DCT) is taken on the horizontal rows of each of the M stripes. For the k^{th} row and j^{th} stripe, we then have DCT coefficients $X_{i,j}(k)$ for $i = 1, \cdots, n$, $j = 1, \cdots, M$, and $k = 1, \cdots, N$.

The DPCM structure operates vertically in the transformed image (based on index k). The first row ($k = 1$) is finely quantized (as $\hat{X}_{i,j}(1)$) and transmitted. In the scalar quantization case, let $\hat{X}_{i,j}(k|k-1)$ be a predicted value of coefficient $X_{i,j}(k)$. Then for each i, j the difference signal

$$E_{i,j}(k) = X_{i,j}(k) - \hat{X}_{i,j}(k|k-1), \text{ for } k = 2, \cdots, N \tag{18}$$

is formed, quantized as $\hat{E}_{i,j}(k)$ and transmitted, and used to form the DPCM output

$$\hat{X}_{i,j}(k) = \hat{E}_{i,j}(k) + \hat{X}_{i,j}(k|k-1). \tag{19}$$

Many different predictors are possible. In the present work the fixed first–order predictor

$$\hat{X}_{i,j}(k|k-1) = \alpha\hat{X}_{i,j}(k) \tag{20}$$

was used exclusively.

Vector quantization is introduced by collecting several $X_{i,j}(k)$ values as a vector. To approximate the i.i.d. assumption implicit in the PVQ formulation, vector $\mathbf{X}_{i,l}(k)$ is formed by taking like coefficients from every p^{th} stripe, as

$$\mathbf{X}_{i,l}(k) = (X_{i,l}(k),\ X_{i,l+p}(k),\ X_{i,l+p(L-1)}(k)), \tag{21}$$

where $p = ML$ is an integer, L is the vector dimension, and $l = 1, \ldots, p$ determines the initial stripe. That is, the components of $\mathbf{X}_{i,l}(k)$ consist of like coefficients (a common value of i) from nonadjacent stripes, with p determining how many stripes are skipped between coefficients.

The vector encoding scheme is shown in Figure 3. Vector $\mathbf{X}_{i,l}(k)$ is normalized by radial parameter

$$r_{i,l}(k) = \|\mathbf{X}_{i,l}(k)\|_1 \tag{22}$$

as

$$\mathbf{Y}_{i,l}(k) = \frac{\mathbf{X}_{i,l}(k)}{r_{i,l}(k)}. \tag{23}$$

Parameter $r_{i,l}(k)$ is then DPCM encoded with a fixed (optimum) scalar quantizer with output $\hat{r}_{i,l}(k)$. Vector $\mathbf{Y}_{i,l}(k)$ is differentially encoded (analogous to (18)–(20)) by pyramid vector quantizing

$$\mathbf{E}_{i,l}(k) = \mathbf{Y}_{i,l}(k) - \alpha \mathbf{Y}_{i,l}(k-1).$$

The PVQ output, $\hat{\mathbf{E}}_{i,l}(k)$, is then transmitted and used to form

$$\hat{\mathbf{y}}_{i,l}(k) = \hat{\mathbf{E}}_{i,l}(k) + \alpha \hat{\mathbf{Y}}_{i,l}(k-1).$$

The overall reproduction vector is then

$$\hat{\mathbf{X}}_{i,l}(k) = \hat{r}_{i,l}(k) \hat{\mathbf{Y}}_{i,l}(k).$$

The coder output vectors are then reconfigured in the proper stripes, and the inverse DCT used to produce the reconstructed image.

IV. PERFORMANCE

The performance of the PVQ–based hybrid transform coding scheme was evaluated by encoding several images at average rates of $\frac{1}{2}$, 1, 2, and 3 bits per pixel. For each case, the analogous scalar quantizer hybrid coding was also performed. Rate assignments for the various coefficients were determined according to [1] and were used for both the PVQ and scalar quantizer coders. In all cases the quantizers were fixed and no adaptive quantization or prediction was considered.

For simplicity, offset p in (21) was set to unity and the combinations of 16–point DCT and 16–dimensional quantization, and 8–point DCT and 32–dimensional quantization examined (for 256 $\times$ 256 images). The peak signal–to–noise ratio (PSNR) performance of both the scalar quantization and PVQ coders is shown in Table 1.

As evidenced by Table 1, the PVQ approach offers significant improvement in PSNR, and this improvement is quite noticeable in image quality at the lower bit rates. The scalar quantization coding generally suffered a loss in PSNR when the block size used in the cosine transform was reduced from 16 (for $L = 16$) to 8 (for $L = 32$). For the vector quantization approach, the opposite was true, since there is considerable improvement in PVQ performance when the vector dimension is increased from 16 to 32.

REFERENCES

[1] W. K. Pratt, *Digital Image Processing.* New York: Wiley, 1978.

[2] A. Habibi, "Hybrid coding of pictorial data," *IEEE Trans. Commun.*, vol. COM–22, pp 614–624, May 1974.

[3] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C–23, pp. 90–93, Jan 1974.

[4] P. A. Wintz and A. J. Kurtenbach, "Waveform error control in PCM telemetry," *IEEE Trans. Inform. Th.*, vol. IT–14, No. 5, pp. 650–661, Sept. 1968.

[5] A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Inform. Th.*, vol. IT–28, pp. 157–166, March 1982.

[6] J. H. Conway and N. J. A. Sloane, "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Trans. Inform. Th.*, Vol. IT–28, pp. 211–226, March 1982.

[7] K. Sayood, J. D. Gibson, and M. C. Rost, "An algorithm for uniform vector quantizer design," *IEEE Trans. Inform. Th.*, to appear.

[8] Y. L. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM–28, pp. 84–95, Jan. 1980.

[9] A. Gersho and B. Ramamurthi, "Image coding using vector quantization," *Proc. IEEE Intern. Conf. Acoustics, Speech, and Sig. Process*, vol. 1, pp 428–431, April 1982.

[10] R. L. Baker and R. M. Gray, "Image compression using non–adaptive spatial vector quantization," *Conf. Rec. 16th Asilomar Conf. on Circuits, Systems, and Computers*, Oct. 1982.

[11] R. C. Reininger and J. D. Gibson, "Distributions of the two–dimensional DCT coefficients for images," *IEEE Trans. Commun.*, vol. COM–31, pp. 835–839, June 1983.

[12] T. R. Fischer, "A laplacian vector quantizer," *Proc. 21st Annual Allerton Conf.*, Univ. of Ill. pp. 501–510, Oct. 1983.

[13] T. R. Fischer, "Geometric source coding and vector quantization," submitted to *IEEE Trans. Inform. Th.*.

[14] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, and Sig. Process.*, Vol. ASSP–28, No. 5, pp. 562–574, Oct. 1980.

[15] M. J. Sabin and R. M. Gray, "Product code vector quantizers for waveform and voice coding," *IEEE Trans. Acoust., Speech, and Sig. Process*, vol. ASSP–32, No. 3, pp. 474–488, June 1984.

[16] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Th.*, vol. IT–6, pp. 7-12, March 1960.

[17] J. J. Y. Huang and P. M. Schultheiss, "Block quantization of correlated Gaussian random variables," *IEEE Trans. Commun. Syst.*, vol. CS–11, pp. 289–296, Sept. 1963.

[18] L. M. Goodman, "Optimum rate allocation for encoding sets of analog messages," *Proc. IEEE*, vol. 53, pp.1776-1777, Nov. 1965.

[19] The University of Southern California Image Processing Institute, "Image data base," Vol. 5.

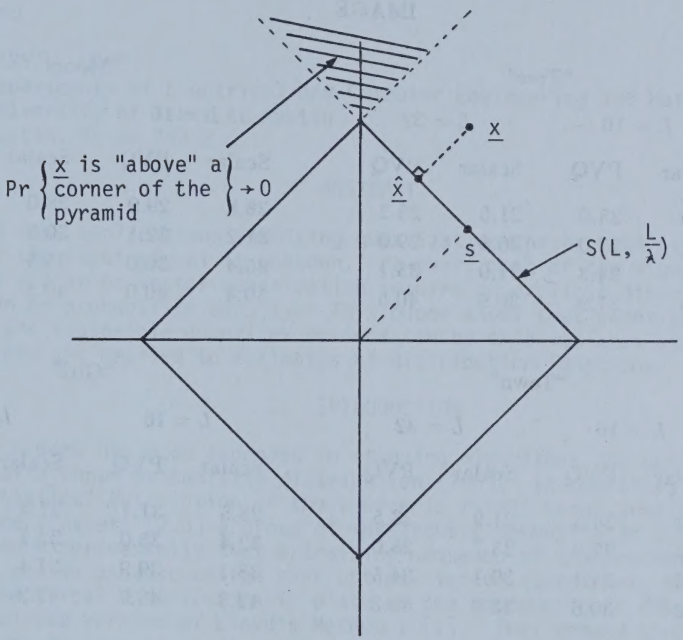


Figure 1. Finding the closest vector on the pyramid.

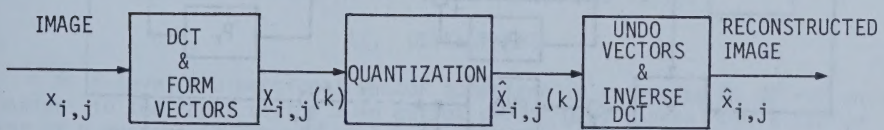


Figure 2. Hybrid coding of the image.

TABLE 1

Peak signal-to-noise ratios (in dB) for scalar quantization and PVQ hybrid/transform coding. Images from [19].

IMAGE								
	"Tree"				"Moon"			
	$L = 16$		$L = 32$		$L = 16$		$L = 32$	
Rate	Scalar	PVQ	Scalar	PVQ	Scalar	PVQ	Scalar	PVQ
0.5	23.0	25.0	21.5	25.2	28.8	29.9	28.0	29.8
1	27.1	29.0	26.8	29.0	31.2	32.1	30.9	32.1
2	32.4	34.3	31.9	35.1	35.4	36.0	35.5	37.2
3	36.6	37.9	36.9	40.5	39.4	40.0	40.1	42.4
	"Town"				"Girl"			
	$L = 16$		$L = 32$		$L = 16$		$L = 32$	
Rate	Scalar	PVQ	Scalar	PVQ	Scalar	PVQ	Scalar	PVQ
0.5	23.4	24.3	21.9	23.5	28.5	31.1	27.3	31.0
1	27.3	27.9	25.7	28.0	32.4	35.0	32.1	35.1
2	33.0	34.1	30.1	34.5	38.1	39.9	37.4	40.9
3	38.2	39.6	33.0	40.3	42.3	43.7	42.2	46.1

$$PSNR \triangleq 10 \log_{10} \left[\frac{255}{(256)^{-2} \sum_{i=1}^{256} \sum_{j=1}^{256} (x(i,j) - \hat{x}(i,j))^2} \right].$$

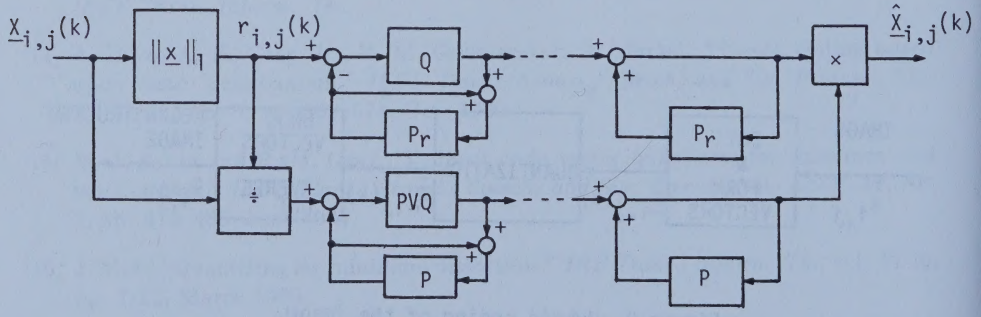


Figure 3. PVQ based encoding of the vector of transform coefficients.

SOME COMMENTS ON THE DESIGN OF QUANTIZERS

EFREN F. ABAYA
Department of Electrical Engineering
University of the Philippines
Diliman, Quezon City 3004, Philippines

and

GARY L. WISE
Departments of Electrical and Computer Engineering and Mathematics
University of Texas at Austin
Austin, Texas 78712

ABSTRACT

In many applications involving quantization the probability distribution of the input signal is unknown. However, most of the algorithms for optimal scalar or vector quantization require an explicit distribution function or probability density. This paper shows that under certain conditions reasonable quantizer designs can be expected when standard algorithms are applied to estimates of distribution functions.

I. INTRODUCTION

Much work has been expended in studying algorithms for optimal quantization of a known probability distribution [1-5]. In practice, however, the statistical description of the source is rarely known precisely. In some recent papers [2,3] a group of researchers demonstrated both theoretically and experimentally that a training sequence of independent or ergodic samples can be used to design near optimal vector quantizers, by using the sample empirical distribution in place of the unknown input distribution in a generalized version of Lloyd's Method I [1]. They showed that under some conditions the quantizer designed for a "long" training sequence approximates closely the output levels and performance of the optimal quantizer for the true (unknown) distribution. The same kind of reasoning should hold for any design algorithm. If the input distribution F is not known, then we can form an estimate $\hat{F}_n$ based on n observations of the input signal. As n becomes large, we expect a reasonable estimate to converge to the true distribution F. Intuitively, then, an optimal quantizer designed for $\hat{F}_n$, and the resulting distortion, should closely approximate those of an optimal quantizer for F. In this paper we will establish properties of an estimator $\hat{F}_n$ so that this kind of reasoning will be valid.

II. DEVELOPMENT

An N-level k-dimensional vector quantizer is a mapping $Q: \mathbb{R}^k \to \mathbb{R}^k$ which assigns to the input vector x an output vector Q(x) chosen from a finite set of N vectors $\{y_i: y_i \in \mathbb{R}^k, i=1,2,\ldots,N\}$. The distortion incurred in quantizing a k dimensional random variable X having a probability distribution function F is expressed by

$$D(Q,F) = \int C_0(||x-Q(x)||)\, dF(x) \tag{1}$$

where $||\cdot||$ is the usual Euclidean norm on $\mathbb{R}^k$ and where all integrals, unless noted otherwise, are over $\mathbb{R}^k$. We will take the cost function $C_0(t)$ to be nonnegative, nondecreasing on $[0,\infty)$ and lower semi-continuous. It has been shown previously that optimal quantizers minimizing (1) exist for

all probability distributions F and all k and N [6,7]. This type of distortion function subsumes most of the popular error criteria used in scalar and vector quantization. When optimal quantizers are being considered, there is no loss of generality in assuming the "nearest neighbor" assignment rule: $Q(x)$ is that member of the set which is nearest to x in Euclidean norm, with ties being broken by an arbitrary preassigned method. This rule will be adopted throughout the rest of the paper. Thus a quantizer is completely represented by its set of output vectors.

We will say that a sequence of N-level quantizers $\{Q_n\}$ converges weakly to the quantizer Q if $Q_n(x) \to Q(x)$ at all continuity points x of Q.

Let F_n and F be k-variate probability distribution functions. The sequence $\{F_n\}$ is said to converge weakly to F (written $F_n \xrightarrow{W} F$) if $F_n(x) \to F(x)$ at every continuity point x of F. We say that $\{F_n\}$ converges setwise to F (denoted $F_n \xrightarrow{S} F$) if

$$\lim_{n\to\infty} \int_B dF_n(x) = \int_B dF(x)$$

for every Borel subset B of $\mathbb{R}^k$.

The following theorem from [8] and [9] is the main tool used in the investigation. Here k and N are fixed positive integers.

Theorem 1: Assume that $C_0(t)$ is nonnegative and nondecreasing on $[0,\infty)$. Suppose that $C_0(||x-y||)$ is uniformly integrable with respect to a sequence of distribution functions $\{F_n\}$ for every y. Let Q_n be an optimal N-level quantizer for F_n. If either (a) $C_0(t)$ is continuous and $F_n \xrightarrow{W} F$; or (b) $C_0(t)$ is lower semicontinuous and $F_n \xrightarrow{S} F$, then every convergent subsequence of $\{Q_n\}$ converges weakly to an optimal quantizer for F. Such a subsequence exists unless $C_0(||x||)$ is constant with F-probability 1. Moreover, $D(Q_n,F_n)$ converges to the optimal mean distortion for quantizing F with N levels.

In Theorem 1, convergence of the sequence of optimal quantizers $\{Q_n\}$ cannot be asserted. However, $\{Q_n\}$ can be completely partitioned into convergent subsequences whose limits are optimal quantizers for F.

In Theorem 1, F_n can be any sequence of distribution functions. We are interested in having F_n be an estimate $\hat{F}_n$ constructed from n observations of F. The principal consideration in applying the theorem is in showing uniform integrability. For sequences of estimates this becomes

$$\lim_{n\to\infty} \int C_0(||x-y||)\, d\hat{F}_n(x) = \int C_0(||x-y||)\, dF(x) \qquad (2)$$

for each $y \in \mathbb{R}^k$; for almost all training sequences. Before proceeding further, we make a useful simplification of (2). This equation requires us to find a set of sample sequences $(X_1,X_2,\ldots)$ on which the integrals converge for every y in $\mathbb{R}^k$. What we will be using primarily is the Strong Law of Large Numbers, which yields the conclusion:

$$\lim_{n\to\infty} \int C_0(||x-y||)\, d\hat{F}_n(x) = \int C_0(||x-y||)\, dF(x) \qquad (3)$$

for almost all training sequences; for each $y \in \mathbb{R}^k$. This last equation says that for a given y, there is a set $\Lambda(y)$ of sample sequences $(X_1,X_2,\ldots)$ having probability one for which the integrals converge. Obviously, (2) implies (3). However, it is shown in the Appendix that (3) implies (2). Thus (2) and (3) are equivalent. Hence if we can show (3), we may use Theorem 1 to get the results we want. In the following examples, we will

show how this can be done in a number of situations.

Empirical Distributions. Let F be the unknown k-variate distribution which we wish to (vector) quantize. If we take n independent samples $X_1, X_2, \ldots, X_n$, then the empirical distribution function is $\hat{F}_n(x) = n^{-1}(\#X_i \leq x)$, where the inequality is taken component by component. This is a simple nonparametric estimator which by the Strong Law of Large Numbers converges setwise to F for almost all sample sequences as $n \to \infty$. Take C_0 to be nonnegative, nondecreasing, nonconstant with respect to F, and lower semi-continuous. The demonstration of uniform integrability (3) is a simple application of the Strong Law of Large Numbers. We have for each y

$$\lim_{n \to \infty} \int C_0(\|x-y\|)\, d\hat{F}_n(x) = \int C_0(\|x-y\|)\, dF \quad \text{wpl},$$

provided that the last integral is finite. According to Theorem 1, then, we may quantize the $\hat{F}_n$'s, using any available method that yields an optimal solution and be assured that the resulting quantizers Q_n converge weakly to an optimal quantizer for F. This generalizes the analysis of [2,3] to other algorithms besides the extension of Lloyd's Method I. Instead of assuming the training set to consist of independent samples, we may make an ergodic (or block ergodic) assumption. This seems to be a useful assumption regarding information sources. In the development, the Strong Law of Large Numbers can be replaced by an ergodic theorem [10] and all of the conclusions reached above will remain valid.

For the rest of the paper, we will restrict our attention to distributions having a density, which we denote by f. Also, except for a portion of the last example, we will specialize to r-th power distortions, which have been widely used and studied. In this case, a natural procedure would be to use some density estimator $\hat{f}_n$ based on a training sample $X_1, X_2, \ldots$ in the quantization algorithm. Then (3) becomes

$$\lim_{n \to \infty} \int \|x-y\|^r \hat{f}_n(x)dx = \int \|x-y\|^r f(x)dx \tag{4}$$

for almost all training sequences; for each $y \in \mathbb{R}^k$. This equation says that if $\hat{f}_n$ and f are to behave in almost the same manner using r-th power quantizers, then they should have nearly the same r-th moments about arbitrary points y. Viewed differently, the quantity

$$\int \|x-y\|^r \hat{f}_n(x)dx$$

is an estimator of the r-th moment about y of the density f. Theorem 1 requires this estimator to be strongly consistent, i.e., to converge almost surely to the true moment of f. In addition, of course, we want the distribution associated with $\hat{f}_n$ to converge weakly to the distribution of f. This is satisfied if we have

$$\lim_{n \to \infty} \hat{f}_n(x) = f(x) \tag{5}$$

for almost all x; for almost all training sequences. The discussion thus far can be summarized by saying that an estimator $\hat{f}_n$ based on samples can be used in a quantizer design algorithm in place of f if it satisfies the strong consistency conditions (4) and (5). The examples below illustrate this point.

Normal densities. Consider that the density f to be quantized is univariate normal with unknown mean μ and variance $\sigma^2 > 0$. Let $\mathcal{N}(x)$ denote the standard normal density. Then we may write $f(x) = \mathcal{N}((x-\mu)/\sigma)/\sigma$. Let $X_1, X_2, \ldots$ be independent, identically distributed samples from f. Strongly consistent estimates of the unknown parameters μ and σ^2 are, respectively,

the sample mean and variance, $\hat{\mu}_n$ and $\hat{\sigma}_n^2$. Let $\hat{f}_n$ be a normal density with these parameters. This has the strong consistency property (5). Next we will show (4). By a change of variable

$$\int |x-y|^r \hat{f}_n(x)dx = \int |\hat{\sigma}_n x + \hat{\mu}_n - y|^r \mathcal{N}(x)dx.$$

Upon applying the c_r inequality [11, p.157] we have

$$|\hat{\sigma}_n x + \hat{\mu}_n - y|^r \leq c_r |\hat{\sigma}_n|^r |x|^r + c_r |\hat{\mu}_n - y|^r.$$

By the strong consistency of the estimators,

$$\lim_{n\to\infty} \int (c_r |\hat{\sigma}_n|^r |x|^r + c_r |\hat{\mu}_n - y|^r) \mathcal{N}(x)dx = \int (c_r \sigma^r |x|^r + c_r |\mu - y|^r) \mathcal{N}(x)dx \quad \text{wpl.}$$

Thus (4) follows upon invoking a generalized Dominated Convergence Theorem [12, p.89].

The family of normal densities in this example can be replaced with any class of almost everywhere continuous densities parametrized by location and/or scale parameters. We may contemplate other instances in which the unknown parameter is neither a location nor scale parameter, but where the above analysis can be useful. For instance, the exponent p in the generalized Gaussian density $f(x) = K_p \exp(-\gamma |x|^p)$ can be varied to fit many histograms.

In some situations it might be more appropriate to use a nonparametric density estimator. A popular type of estimator is the kernel density estimator introduced by Parzen [13] for univariate densities and generalized to multivariate densities by later authors: $f_n(x) = n^{-1} \sum_{i=1}^{n} h_n^{-k} K((x-X_i)/h_n)$. The kernel $K(x)$ is a probability density function on $\mathbb{R}^k$ and $\{h_n\}$ is a sequence of numbers decreasing to zero. Nadaraya [14] shows for the univariate case that if

$$\begin{aligned} &f(x) \text{ is a uniformly continuous density,} \\ &K(x) \text{ has bounded variation, and} \\ &\sum_{n=1}^{\infty} \exp(-\gamma n h_n^2) \text{ converges for every } \gamma > 0, \end{aligned} \tag{6}$$

then $\hat{f}_n(x) \to f(x)$ uniformly with probability one. The extensions of the result to several dimensions use slightly different sets of assumptions. We give the result of Moore and Yackel [15] as a typical example. If

$$\begin{aligned} &f(x) \text{ is a uniformly continuous density on } \mathbb{R}^k, \\ &K(x) \text{ is a bounded density on } \mathbb{R}^k, \\ &K(x) \text{ has bounded variation, and} \\ &n h_n^{2k} / \log n \to \infty \text{ as } n \to \infty, \end{aligned} \tag{7}$$

then $\hat{f}_n(x) \to f(x)$ uniformly with probability one.

Assuming that (6) or (7) holds, the distribution of $\hat{f}_n$ converges setwise to that of f. In addition, K and $\{h_n\}$ should be chosen appropriately so that (4) is satisfied. Two examples are given below. The first is for scalar quantization and the second for vector quantization.

Compact support. Assume that the unknown density f has compact support (which we may take without loss of generality to be contained in

[-1,1]) and is uniformly continuous on the real line. We wish to quantize this optimally using a cost function C_0 which is nonnegative, nondecreasing and lower semicontinuous, so we also assume that $\int C_0(|x-y|)\,K(x)\,dx < \infty$ for all y. For convenience we take the kernel to be symmetric and unimodal, so it decreases away from the origin. Consider

$$\int_{|x|>a} C_0(|x-y|)\hat{f}_n(x)dx \le 2\int_{x>a} C_0(|x-y|)\frac{1}{h_n}K\left(\frac{x-1}{h_n}\right)dx \qquad \text{wpl.}$$

After a change of variable and simplification, we have

$$\int_{|x|>a} C_0(|x-y|)\hat{f}_n(x)dx \le 2\int_{x>a/h_1} C_0(|h_1x+1-y|)\,K(x)dx \qquad \text{wpl.}$$

It follows that $C_0(|x-y|)$ is uniformly integrable with respect to $\{\hat{f}_n(x)\}$, wpl. Therefore the kernel estimator is a viable basis for designing a scalar quantizer for a density with compact support.

Noncompact support. We can extend the previous analysis to multivariate densities with unbounded support if we restrict attention to r-th power quantizers. A bounded continuous density f(x) whose tails go to 0 as $\|x\|\to\infty$ is uniformly continuous, so none of the common densities are excluded by the assumption (7). Let the cost function be $C_0(t) = t^r$ where temporarily $r = 2p$ is an even integer. We will also make the natural assumptions that $\|x\|$ has finite (2p)-th moments with respect to both densities f and K. Our immediate goal is to show (4) for $r = 2p$. By the c_r-inequality [11, p.157] we have

$$\|x-y\|^{2p}\hat{f}_n(x) \le c_p \sum_{j=1}^{k} n^{-1}\sum_{i=1}^{n}\left(x^{(j)}-y^{(j)}\right)^{2p} K\left(\frac{x-X_i}{h_n}\right)h_n^{-k}, \qquad (8)$$

where the superscript denotes the j-th component of the k-dimensional vector. Note that the right-hand side converges pointwise as $n\to\infty$ to

$$c_p \sum_{j=1}^{k}\left(x^{(j)}-y^{(j)}\right)^{2p} f(x).$$

If the right-hand side of (8) were integrated, we get k terms, a typical one of which looks like the following if we ignore the constant c_p:

$$m_{j,n} = n^{-1}\sum_{i=1}^{n}\int_{\mathbb{R}^k}\left[h_n x^{(j)} + X_i^{(j)} - y^{(j)}\right]^{2p} K(x)dx.$$

We can use the binomial theorem to expand the last integrand. After rearranging the sums we get

$$m_{j,n} = n^{-1}\sum_{i=1}^{n}\left(X_i^{(j)}-y^{(j)}\right)^{2p} + \sum_{\ell=0}^{2p-1}\binom{2p}{\ell}h_n^{2p-\ell}$$
$$\cdot\left\{n^{-1}\sum_{i=1}^{n}\left(X_i^{(j)}-y^{(j)}\right)^{\ell}\int_{\mathbb{R}^k}[x^{(j)}]^{2p-\ell}\,K(x)dx\right\}.$$

The terms in the summation from $\ell = 0$ to $\ell = 2p-1$ are all multiplied by a power of h_n, which decreases to 0, so that in the limit, the second group of terms above is zero. Using the Strong Law of Large Numbers on the first term gives

$$\lim_{n\to\infty} m_{j,n} = \int_{\mathbb{R}^k}\left(x^{(j)}-y^{(j)}\right)^{2p} f(x)dx \qquad \text{wpl.}$$

This is true for all j in (8). Therefore

$$\lim_{n\to\infty} \int c_p \sum_{j=1}^{k} \left(x^{(j)}-y^{(j)}\right)^{2p} \hat{f}_n(x)dx = \int c_p \sum_{j=1}^{k} \left(x^{(j)}-y^{(j)}\right)^{2p} f(x)dx \quad \text{wpl.}$$

Since $||x-y||^{2p}\hat{f}_n(x)$ converges to $||x-y||^{2p}f(x)$ wpl, we may use a generalization of the Dominated Convergence Theorem [12, p.89] to conclude that (4) holds for $r=2p$. Therefore we have uniform integrability for the cost function with r an even integer. In general, for any positive r, we can let 2p be the smallest even integer for which $r \le 2p$. Then the above implies that [16, p.252]

$$\lim_{n\to\infty} \int ||x-y||^r \hat{f}_n(x)dx = \int ||x-y||^r f(x)dx \qquad \text{wpl.} \tag{9}$$

Thus the kernel estimator $\hat{f}_n$ can be used to design a quantizer for f.

Another extension of the analysis is possible. Consider a cost function $C_0(t)$ which grows polynomially fast, i.e., there exist α, β and r so that

$$0 \le C_0(t) \le \alpha t^r + \beta. \tag{10}$$

Since

$$\lim_{n\to\infty} C_0(||x-y||)\hat{f}_n(x) = C_0(||x-y||)f(x) \qquad \text{wpl,}$$

(9) and (10) imply, through the generalization of the Dominated Convergence Theorem [12, p.89], that

$$\lim_{n\to\infty} \int C_0(||x-y||)\hat{f}_n(x)dx = \int C_0(||x-y||)f(x)dx \qquad \text{wpl.}$$

So if C_0 is nondecreasing and lower semicontinuous, then Theorem 1 also applies, and $\hat{f}_n$ can be used to design a quantizer with the cost function C_0. Eq. (10) is equivalent to having a $\lambda > 0$ so that

$$C_0(2t) \le \lambda C_0(t) \qquad \text{for all t.} \tag{11}$$

This is a useful form for a cost function because (11) and $\int C_0(||x||)dF < \infty$ imply that $\int C_0(||x-y||)dF < \infty$ for all vectors y.

As an application of this example, consider Max's algorithm [4] (also called Lloyd's Method II [1] or the Lloyd-Max algorithm). Some aspects of this algorithm were mentioned earlier, and we noted that as it has been described in the literature, the algorithm requires the existence of a known density function. It does not seem to be readily adapted for quantizing an empirical distribution function. However, with the approach outlined in this paper, it is possible to use density estimates as the input to Max's algorithm and get strongly consistent estimates of the optimal output levels and breakpoints. Thus the algorithm can be (indirectly) driven by a training sequence of independent observations.

APPENDIX

In this appendix we show that (3) implies (2). Eq. (3) says that for a given $y \in \mathbb{R}^k$, there is a set $\Lambda(y)$ of sample sequences $(X_1, X_2, \ldots)$ having probability one for which the integrals converge. Since $\mathbb{R}^k$ is separable,

it is possible to find a dense and countable set $\{y_j\}$ for which this convergence holds with probability one. In fact, for any countable subset of $\mathbb{R}^k$,

$$\Lambda = \bigcap_{j=1}^{\infty} \Lambda(y_j)$$

is a set of probability one. Now, for a given y in (2), we may find a finite subset A of the y_j's so that

$$C_0(||x-y||) \leq \sum_{y_j \in A} C_0(||x-y_j||).$$

This involves constructing a cube around y and picking the y_j outside the cube so that $||x-y|| \leq ||x-y_j||$ for at least one y_j. Then

$$\sup_n \int_{||x|| > a} C_0(||x-y||)\, d\hat{F}_n(x)$$

$$\leq \sum_{y_j \in A} \sup_n \int_{||x|| > a} C_0(||x-y_j||)\, d\hat{F}_n(x).$$

For each y_j and each sample sequence in Λ, the right-hand side can be made arbitrarily small by appropriately choosing a. Therefore $C_0(||x-y||)$ is uniformly integrable for every y, with respect to every $\{\hat{F}_n\}$ arising from the set Λ. This is equivalent to (2). Therefore, (3) implies (2).

ACKNOWLEDGEMENT

This research was supported by the Air Force Office of Scientific Research under Grant AFOSR-81-0047.

REFERENCES

[1] S. P. Lloyd, "Least squares quantization in PCM," Bell Telephone Labs Memorandum, Murray Hill, NJ, 1957, reprinted in *IEEE Trans. Information Theory*, IT-28, no. 2, pp. 129-137, March 1982.

[2] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Communication*, COM-28, no. 1, pp. 84-95, Jan. 1980.

[3] R. M. Gray, J. C. Kieffer and Y. Linde, "Locally optimal block quantizer design," *Information Control*, 45, no. 2, pp. 178-198, May 1980.

[4] J. Max, "Quantizing for minimum distortion," *IRE Trans. Information Theory*, IT-6, no. 1, pp. 7-12, March 1960.

[5] D. K. Sharma, "Design of absolutely optimal quantizers for a wide class of distortion measures," *IEEE Trans. Information Theory*, IT-24, no. 6, pp. 693-702, Nov. 1978.

[6] E. F. Abaya and G. L. Wise, "Some notes on optimal quantization," *Proc. International Conference on Communications*, vol. 2, pp. 30.7.1-30.7.5, June 14-17, 1981.

[7] E. F. Abaya and G. L. Wise, "On the existence of optimal quantizers," *IEEE Trans. Information Theory*, IT-28, pp. 937-940, Nov. 1982.

[8] E. F. Abaya and G. L. Wise, "Convergence of vector quantizers with application to the design of optimal quantizers," *Proc. 19th Annual Allerton Conference on Communication, Control, and Computing*, pp. 79-88, Sept. 30-Oct. 2, 1981.

[9] E. F. Abaya and G. L. Wise, "Convergence of vector quantizers with applications to optimal quantization," *SIAM J. Applied Mathematics*, 44, pp. 183-189, Feb. 1984.

[10] L. Breiman, "The individual ergodic theorem of information theory," *Ann. Math. Stat.*, 28, pp. 809-811, Sept. 1957.

[11] M. Loeve, *Probability Theory 1*, 4th Edition, Springer-Verlag, New York, 1977.

[12] H. L. Royden, *Real Analysis*, 2nd Ed., Macmillan Publ. Co., New York, 1968.

[13] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat.*, 33, pp. 1065-1076, Sept. 1962.

[14] E. A. Nadaraya, "On non-parametric estimates of density functions and regression curves," *Th. Prob. Appl.*, vol. X, pp. 186-190, 1965.

[15] D. S. Moore and J. W. Yackel, "Consistency properties of nearest neighbor density function estimators," *Ann. Stat.*, 5, pp. 143-154, Jan. 1977.

[16] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. II, 2nd Ed., John Wiley and Sons, New York, 1971.

ON SELF-TUNING TO THE OPTIMAL CONTROLLER

P. R. KUMAR
Department of Mathematics and Computer Science
University of Maryland Baltimore County
5401 Wilkens Avenue
Baltimore, Maryland 21228, U.S.A.

SUMMARY

The question of whether some adaptive controllers tune themselves automatically to the optimal feedback controller will be examined.

First we consider an adaptive control law consisting of a stochastic approximation (or stochastic gradient) parameter estimator followed by a minimum variance control law. Under some conditions it is proved that

i) the parameter estimates coverage to a random multiple of the true parameter,
ii) and so, the adaptive controller self-tunes to the true minimum variance controller,
iii) even though the standard persistency of excitation condition is not satisfied.

For details, the reader is referred to (1,2).

Second, we consider an adaptive control law consisting of a stochastic approximation (or stochastic gradient) parameter estimator followed by a linear-quadratic optimal control law. Ljung's O.D.E.'s are examined, and it is shown that though the O.D.E.'s may coverge, they will not generally converge to the optimal linear-quadratic regulator.

In fact, convergence to the optimal control law in the minimum variance case rests on a fortuitous mathematical coincidence, and it is doubtful whether one can get similar results in general. For details, the reader is referred to (3).

References:

1. A. Becker, P.R. Kumar, C.Z. Wei, "Adaptive Control with the Stochastic Approximation Algorithm", to appear in IEEE Transactions on Automatic Control, March 1985.
2. P.R. Kumar, "A Survey of Some Results in Stochastic Adaptive Control", to appear in SIAM Journal on Control and Optimization.
3. W. Lin, P.R. Kumar and T.I. Seidman, "Will the Self Tuning Approach Work for General Cost Criteria?"

ROBUST CONTROL OF LINEAR STOCHASTIC SYSTEMS WITH UNCERTAIN DYNAMICS: A GAME-THEORETIC APPROACH

ELIZABETH RYAN-SCHUCK & MAX MINTZ
Department of Systems Engineering
University of Pennsylvania
Philadelphia, PA

INTRODUCTION & SUMMARY

This paper addresses the problem of controlling a discrete-time finite-dimensional linear stochastic system with uncertain dynamics. Here, the uncertain dynamics are represented by a given uncertainty class -- a specified set of dynamic systems. Our goal is the determination of a linear control law which is robust with respect to this uncertainty class -- as measured by the expected value of a given quadratic performance criterion. In this problem formulation we consider several possible information patterns for the controller: (1) partial state information in additive white noise with limited/unlimited memory capacity; and (2) complete noise-free state information with limited/unlimited memory capacity.

Recent research in robust filtering and control for linear systems with quadratic performance criteria has appeared in [1,2,3]. The robust control results obtained in [1] pertain to a steady-state problem where the uncertainty in the system description is restricted to the spectral density matrices of the white process noise and white observation noise. The robust control results obtained in [2] pertain to a time-varying dynamic system where the uncertainty in the system description is again restricted to the process and observation noise intensity matrices. The steady-state solution obtained in [1] is characterized by a saddle-point in pure strategies, i.e, there is an optimal nonrandomized choice of controller-estimator, and a corresponding pair of worst-case spectral density matrices. Similarly, the robust solution for the nonstationary problem obtained in [2] is characterized by a saddle-point in pure strategies. The analyses in [1,2] do not include the case where the system "F-matrix" is uncertain. A robust filtering-prediction problem, which corresponds to the present robust control problem -- in terms of its underlying uncertainty class, has been considered in [3]. This uncertainty class includes m-th order time-varying discrete-time systems with uncertain dynamics, uncertain nonstationary input and observation noise intensity matrices, and uncertain initial state covariance. The robust filter-predictor obtained in [3] can be realized by an augmented qm-th order Kalman filter. Further, the worst-case probability distribution for the uncertain system parameters is a discrete q-point least favorable prior distribution.

The primary contributions of this paper are: (1) a delineation of the salient features of the underlying minimax decision problem; (2) a set of sufficient conditions for a saddle-point solution which is characterized by a pure strategy for the control system designer and a mixed strategy for the designer's opponent -- nature; and (3) explicit saddle-point solutions for several illustrative first-order examples.

In the remainder of this paper we: (1) formally define our robust control paradigms; (2) examine the structure of a game-theoretic kernel; (3) obtain a saddle-point theorem; (4) analyze the behavior of the minimax and maximin strategies; (5) obtain numerical solutions for illustrative examples; and (6) consider directions for further research.

ROBUST CONTROL PARADIGMS

In this section we delineate a class of mathematical decision models which define the robust control problems addressed in this paper.

Preliminary Notation: The following notation applies unless otherwise noted. Lower-case symbols, with or without underbars or subscripts, denote scalar or vector quantities. Upper-case symbols, with or without underbars or subscripts, denote matrices or subsets of a Euclidean space. We adopt the following specific conventions:

E^n	the n-dimensional Euclidean space.
E( )	the expected value operator.
tr( )	the trace operator.
T^0	the finite index set {1, 2, 3, ..., N}.
T^1	the positive integers {1, 2, 3, ... }.
A'	the transpose of the matrix A.
u'	the transpose of the vector u.
con(S)	the convex hull of the set S.

Structural Definitions: In this paper an uncertainty class denotes an indexed collection of models which is assumed to contain the the true dynamic system that we seek to control. Further, we shall assume that the given uncertainty class can be represented by an equivalent set of parameter matrices which define the system dynamics, as well as the second-order statistics of the initial state, input process noise, and observation noise. Since any set of matrices with common dimensions, or any Cartesian product of such sets, can be embedded in an appropriate Euclidean space, we shall interpret uncertainty classes in this manner. Thus, a compact uncertainty class denotes a set of parameter matrices which forms a closed bounded subset of a Euclidean space.

A Parametric Uncertainty Class: Define a generic m-dimensional linear dynamic system and associated observables by:

$$x(n+1) = F(n)x(n) + G(n)u(n) + w(n);$$
$$z(k) = H(k)x(k) + v(k). \qquad (1)$$

Assume:

$n, k \in T^0$;

$\{F(n): n \in T^0\} \in \mathcal{F}$;

$\{G(n): n \in T^0\} \in \mathcal{G}$;

$E[w(n)] = 0$, $E[w(i)w(j)'] = \Sigma(i)\delta_{i,j}$, $i,j \in T^0$. $\{\Sigma(n): n \in T^0\} \in \mathcal{S}$;

$E[v(n)] = 0$, $E[v(i)v(j)'] = C(i)\delta_{i,j}$, $i,j \in T^0$, $\{C(n): n \in T^0\} \in \mathcal{C}$;

$$E[x(1)] = 0,\ E[x(1)x(1)'] = M(1),\ M(1) \in M;$$

where: the component uncertainty classes F, G, S, C, and M are given compact Euclidean subsets, the observation matrices $\{H(n): n \in T^o\}$ are known a priori, and $\{u(n): n \in T^o\}$ denotes a sequence of control actions.

Finally, we define the overall uncertainty class Y by a Euclidean embedding of the Cartesian product:

$$Y = F \times G \times S \times C \times M.$$

Permissible Control Strategies: We consider the following class of linear nonanticipatory control laws:

$$u(n) = \sum_{k=n-s+1}^{n} A(n,k)z(k); \tag{2}$$

where: $n \in T^o$, and s denotes the memory capacity of the control law. The extreme cases $s = 1$, and $s = n$ are of particular interest. When $s = 1$, we say the control law has one-stage or minimal memory. When $s = n$, we say the control law has full or maximal memory.

Since each linear control law (2) is completely parameterized by an associated weighting sequence of gain matrices, it will be useful to denote these by $\underline{A} = \{A(n,k): n \in T^o,\ k = n-s+1,\ldots,n\}$.

A Quadratic Performance Measure: Let J denote the expected value of the quadratic performance function L:

$$J = E[L];$$

$$L = x(N+1)'Q(N+1)x(N+1) + \sum_{n=1}^{N} [x(n)'Q(n)x(n) + u(n)'R(n)u(n)]; \tag{3}$$

where: $\{Q(n) \geq \underline{0}: n = 1,\ldots,N+1\}$, and $\{R(n) > \underline{0}: n = 1,\ldots,N\}$ denote given sequences of symmetric weighting matrices, and $E[\,]$ denotes the expectation with respect to the random variables $\{x(1), w(n), v(n): n \in T^o\}$.

Remark: The value of J is a function of the gain sequence $\underline{A}$ selected by the designer, and the unknown system parameters $y \in Y$ selected by nature.

The following definition is based on the given dynamic stochastic models, control strategies, and performance measure delineated by (1)-(3).

Definition 1: The control law u^o (2) with associated gain sequence $\underline{A}^o$ is a (nonrandomized) robust linear control law, if for all $\underline{A}$:

$$\max_{y \in Y} J(\underline{A}^o, y) \leq \max_{y \in Y} J(\underline{A}, y). \tag{4}$$

Paradigm 1: Obtain a robust linear control law for the decision problem defined by (1)-(4).

A Minimum Variance Performance Measure: Let J' denote the expected value of the quadratic performance function L':

$$J' = E[L'];$$

$$L' = \sum_{n=1}^{N+1} x(n)'Q(n)x(n); \qquad (3')$$

where: $\{Q(n) \geq 0: n = 1,\ldots,N+1\}$ denotes a sequence of symmetric weighting matrices, and E[] denotes the expectation with respect to the random variables $\{x(1), w(n), v(n): n \in T^0\}$.

Remark: Definition 1 can be extended to include the performance measure J'. We will refer to controls satisfying this extended definition as robust in the context of minimax process variance.

Paradigm 2: Obtain a robust (minimax variance) linear control law for the decision problem defined by (1), (2), (3'), and (4), where J' replaces J in (4).

Paradigms 3 and 4 embrace the infinite interval steady-state versions of Paradigms 1 and 2. Here, we assume that T^1 replaces T^0 in (1), and the parameter matrices F, G, H, C, and Σ are time-invariant. However, in order to make Paradigms 3 and 4 meaningful, we will need to invoke the following definition and structural assumption.

Definition 2: The uncertainty class of dynamic systems (1) is uniformly asymptotically stabilizable, if there exists at least one choice of gain sequence $\underline{A} = \{A(n,k): 1 \leq k \leq n, n \in T^1\}$ such that:

$$\sup_{y \in Y} \lim_{N \to \infty} (1/N) \sum_{n=1}^{N} E[||x(n)||^2] < \infty .$$

Assumption 1: The (time-invariant) uncertainty class (1) is uniformly asymptotically stabilizable.

Paradigm 3: Obtain a robust linear control law for the steady-state version of Paradigm 1. Here J becomes:

$$J = \lim_{n \to \infty} E[x(n)'Qx(n) + u(n)'Ru(n)].$$

Paradigm 4: Obtain a robust linear control law for the steady-state version of Paradigm 2. Here, J' becomes:

$$J' = \lim_{n \to \infty} E[x(n)'Qx(n)].$$

Remark: Without Assumption 1 or its equivalent, the uncertainty class would be too large for robust control techniques to be useful, and the resulting minimax cost would be unbounded in Paradigms 3 and 4.

A GAME-THEORETIC APPROACH

In this section we examine the structure of a game-theoretic kernel, obtain a saddle-point theorem, and analyze the general behavior of the minimax and maximin strategies.

A Game-Theoretic Kernel: The solutions to the minimax problems posed in Paradigms 1-4 can be obtained by individual game-theoretic analyses. However, in order to achieve a brief exposition we will focus our primary attention on the game-theoretic aspects of Paradigm 1. A detailed treatment of the remaining paradigms appears in [4].

We begin our analysis of Paradigm 1 with the following sequence of remarks, definitions, and observations.

Remark: For each $y \in Y$ there exists a minimizing gain sequence $\underline{A}(y)$ for $J(\underline{A},y)$ (3). Further, a minimizing linear control law $u(n|y)$ can be realized by the well-known optimal controller - state estimator solution [5]:

$$
\begin{aligned}
&u(n) = -\pi(n)\hat{x}(n);\\
&\pi(n) = D(n)^{-1}G(n)'K(n+1)F(n);\\
&D(n) = R(n)+G(n)'K(n+1)G(n);\\
&K(n) = F(n)'[K(n+1)-K(n+1)G(n)D(n)^{-1}G(n)'K(n+1)]F(n)+Q(n);\\
&K(N+1) = Q(N+1);
\end{aligned}
\tag{5a}
$$

$$
\begin{aligned}
&\hat{x}(n) = F(n-1)\hat{x}(n-1)+G(n-1)u(n-1)+P(n)H(n)'C(n)^{-1}r(n);\\
&r(n) = z(n)-H(n)[F(n-1)\hat{x}(n-1)+G(n-1)u(n-1)];\\
&\hat{x}(1) = P(1)H(1)'C(1)^{-1}z(1);\\
&P(n) = [I-B(n)H(n)]P^{o}(n);\\
&P^{o}(n) = F(n-1)P(n-1)F(n-1)'+\Sigma(n-1);\\
&B(n) = P^{o}(n)H(n)'[H(n)P^{o}(n)H(n)'+C(n)]^{-1};\\
&P^{o}(1) = M(1).
\end{aligned}
\tag{5b}
$$

Notation: Let $\mathcal{P}_Y$ denote the set of all probability measures μ_Y on the measurable space $(Y,\mathcal{B}(Y))$ -- where $\mathcal{B}(Y)$ denotes the Borel sigma-field of Y. Let $\underline{A}(\mu_Y)$ denote a gain sequence which is a minimum $\underline{A}$-norm solution to:

$$\underline{A}(\mu_Y) = \arg\min_{\underline{A}} E_{\mu_Y}[J(\underline{A},y)].$$

Observation 1: The sequence $\underline{A}(\mu_Y)$ is well-defined since: $J(\underline{A},y)$ (3) is continuous in the pair $(\underline{A},y)$, Y is compact, and

$$\lim_{||A||\to\infty} E_{\mu_Y}[J(\underline{A},y)] = \infty.$$

Definition 3: Let A° denote the smallest compact set containing all gain sequences $\underline{x}(\mu_Y)$, $\mu_Y \in P_Y$. Further, let $A = \text{con}(A^{\circ})$.

Notation: Let P_A denote the set of all probability measures μ_A on the measurable space $(A,\beta(A))$ -- where $\beta(A)$ denotes the Borel sigma-field of A.

Definition 4: Let (J,A,Y) denote an abstract game where:

- $J: A \times Y \rightarrow R$ (3);
- A denotes the space of pure strategies for the minimizing player;
- Y denotes the space of pure strategies for the maximizing player.

Theorem 1: The game (J,A,Y) has a saddle-point solution in mixed strategies, i.e., there exist probability measures μ_A° and μ_Y° such that for all μ_A and μ_Y:

$$E_{\mu_A^{\circ} \times \mu_Y}[J] \leq E_{\mu_A^{\circ} \times \mu_Y^{\circ}}[J] \leq E_{\mu_A \times \mu_Y^{\circ}}[J].$$

Proof: The proof follows directly from a standard saddle-point result [6] by noting that: (1) the kernel J is continuous in the pair $(\underline{A},y)$, and (2) A and Y are compact Euclidean subsets.

Corollary 1: If for each $\mu_Y \in P_Y$,

$$\arg\min_{\underline{A}} E_{\mu_Y}[J]$$

is unique, then μ_A° is one-point, i,e., the minimizing player has an optimal pure strategy $\underline{A}^{\circ}$.

Observation 2: The kernel $J(\underline{A},y)$ (3) is generally not convex in $\underline{A}$ for fixed $y \in Y$.

Remark: However, it can be shown that J is convex in the individual components of $\underline{A}$.

Notation: Let the (n,k)th subcomponent of $\underline{A}$ refer to the matrix $A(n,k)$ (2).

Observation 3: If $y \in Y$ is fixed, and all subcomponents of $\underline{A} \in A$ other than $A(n,k)$ are also fixed, then J becomes a convex function of $A(n,k)$.

Observation 4: The kernel $J(\underline{A},y)$ (3) is generally not convex in y for fixed $\underline{A} \in A$.

Observation 5: If $\underline{A} \in A$ is fixed, and all subcomponents of $y \in Y$ other than $F(n)$ $[G(n)]$ are also fixed, then J becomes a convex function of $F(n)$ $[G(n)]$.

Nomenclature: We denote this limited convex behavior of J (3) -- Observations 3 and 5 -- as directional convexity.

Observation 6: The kernel J is also directionally convex in $\Sigma(n)$, $C(n)$, and $M(1)$.

Remark: The directional convexity of J in the subcomponents of y suggests that a least favorable prior distribution μ_Y^o will assign unit probability to the boundary of Y. Further, if Y is a Cartesian product of the individual subcomponent uncertainty classes, and each subcomponent uncertainty class is a convex polyhedron, then the least favorable prior distribution for y will be a discrete distribution which assigns unit probability to the vertices of Y.

EXAMPLES

In this section we consider explicit saddle-point solutions for several illustrative first-order examples.

Example 1: We consider an uncertain time-invariant first-order system with complete noise free state information and limited controller memory capacity. Here, we seek a saddle-point solution in the sense of Paradigm 3.

Let:

$$x(n+1) = fx(n) + gu(n) + w(n);$$

$$z(k) = x(k); \qquad (6)$$

$$u(n) = -\pi x(n);$$

where: $y = f$, $Y = [a,b]$, $E[w(n)^2] = \sigma$, $g > 0$, and $s = 1$.

In order to assure that (6) is uniformly asymptotically stabilizable, we assume that $b < a + 2$.

The kernel J associated with (6) is:

$$J(\pi,f) = [q + r\pi^2]\sigma/[1 - (f-g\pi)^2]; \qquad (7)$$

where: q and r are positive constants corresponding to Q and R.

Observation 7: The kernel J (7) is continuous in the pair (π,f) in the region: $((b-1)/g,(a+1)/g) \times [a,b]$.

Observation 8: The kernel J (7) is convex in π for fixed f, and convex in f for fixed π.

Remark: Since $J(\pi,b)$ $[J(\pi,a)]$ becomes unbounded when π tends to $(b-1)/g$ $[(a+1)/g]$, we can restrict our attention to a compact action space:

$$A = [e+(b-1)/g, -e+(a+1)/g];$$

for a suitably small $e > 0$.

Observation 9: The game (J,A,Y) has a saddle-point solution which is characterized by a pure strategy for the minimizing player, and a mixed strategy for the maximizing player which randomizes over at most two pure strategies in Y.

Remark: Observation 9 is a consequence of Observations 7 and 8, and a saddle-point result due to Blackwell and Girshick [7].

Theorem 2:

Assume $0 \leq a < b$. Let k° denote the positive steady-state solution to the Riccati equation (5a) specialized to (6). Further, let:

$$\pi^\circ(f) = gfk^\circ/(r+g^2k^\circ);$$

$$\tilde{\pi} = (a+b)/2g;$$

$$m = (q+r\tilde{\pi}^2)_\sigma/(1-(b-g\tilde{\pi})^2);$$

$$\lambda = 1/2 - r(a+b)/2g^2(b-a)m.$$

Case 1. If $\pi^\circ(b) > \tilde{\pi}$, then $\tilde{\pi}$ is a robust control gain, and

$$Pr[f=a] = \lambda,$$

$$Pr[f=b] = 1-\lambda,$$

is a least favorable prior distribution.

Case 2. If $\pi^\circ(b) \leq \tilde{\pi}$, then $\pi^\circ(b)$ is a robust control gain, and

$$Pr[f=b] = 1,$$

is a least favorable prior distribution.

Proof: A proof of Theorem 2 appears in [4].

Remarks: Theorem 2 provides a complete solution to this first order example for the case where $a \geq 0$. The remaining cases ($a < 0$, $b > 0$), and ($b \leq 0$) have similar solutions. We refer the reader to [4] for the details.

Numerical Results:

Interval Bounds		Results				
a	b	$\tilde{\pi}$	$\pi^\circ(b)$	$J(\pi,b)$	E[J]	λ
0	0.500	0.156	0.229	-	0.055	0.321
0.200	0.854	0.329	0.404	-	0.062	0.247
0.466	0.854	0.413	0.404	0.067	-	0
0.680	0.854	0.479	0.404	0.067	-	0

(1). Parameter Values: $r = 1$, $q = 1$, $g = 1.6$, $\sigma = 0.5$.

(2). E[J] denotes the value of the game when $\lambda > 0$, i.e., the maximizing player has an optimal mixed strategy.

(3). $J(\pi,b)$ denotes the value of the game when $\lambda = 0$, i.e., the maximizing player has an optimal pure strategy.

Example 2: We reconsider Example 1 when $r = 0$. This is a Paradigm 4 decision model. Here, the game has a saddle-point solution where the robust control gain is $\tilde{\pi} = (a+b)/2g$, and nature's optimal strategy is an equal mixture of $f = a$, and $f = b$.

Remark: It can be shown in Example 2 -- where nature always randomizes -- that the application of additional memory capacity improves the designer's game-theoretic payoff. A detailed discussion of this issue appears in [4].

FURTHER RESEARCH

This paper has addressed a game-theoretic approach to the design of robust linear control laws for uncertain dynamic systems. In further research we are currently studying: (1) the effect of the memory parameter s on the saddle-point solution, and (2) the relationship between robust controllers and robust filters for a given dynamic system.

REFERENCES

[1] Looze, D. P., Poor, H. V., Vastola, K. S., and Darragh, J. C. (1983). Minimax Control of Linear Stochastic Systems with Noise Uncertainty. IEEE Trans. Automat. Contr. Vol AC-28. 882-888.

[2] Verdú, S., and Poor, H. V., (1984). Minimax Linear Observers and Regulators for Stochastic Systems with Uncertain Second-Order Statistics. IEEE Trans. Automat. Contr. Vol AC-29. 499-511.

[3] Martin, C. J., and Mintz, M. (1983). Robust Filtering and Prediction for Linear Systems with Uncertain Dynamics: A Game-Theoretic Approach. IEEE Trans. Automat. Contr. Vol AC-28. 888-896.

[4] Ryan-Schuck, E. and Mintz, M. (1984). Robust Control of Linear Stochastic Systems with Uncertain Dynamics: A Game-Theoretic Approach. To be submitted for publication.

[5] Bertsekas, D. (1976). Dynamic Programming and Stochastic Control. Academic Press, New York.

[6] Owen, G. (1968). Game Theory. Saunders, Philadelphia.

[7] Blackwell, D. H., and Girshick, M. A. (1954). Theory of Games and Statistical Decisions. Wiley, New York.

ACKNOWLEDGMENT

This work was supported in part by the following grants and contracts: the United States Air Force under Contract F33615-83-C-3000, and Grant AFOSR-82-NM-299; the National Science Foundation under Grant MCS-82-07294.

A NON-LINEAR LEAST SQUARE ALGORITHM FOR PARAMETER IDENTIFICATION

JANOS GERTLER and CAIO FERREIRA
Case Western Reserve University, Cleveland, Ohio

ABSTRACT

A technique is described to identify the parameters, including the deadtime, of single input/single output, linear discrete dynamic systems. The method uses a performance index defined as the sum of squared model errors. This model error performance index yields potentially consistent estimates if the output observation noise has zero mean and is uncorrelated with the process input. Minimization of the expectation of the squared output error is accomplished with a version of Powell's method. Such minimization requires the evaluation of the first and second order derivative of the model output; these are obtained as outputs of ARMA processes. Simulation studies were performed in open-loop and in closed-loop inferential controller configuration, with the deadtime being estimated just as another system parameter.

SYSTEM MODEL

A single variable time-invariant linear discrete system may be represented in general as an n^{th} order difference equation relating the input sequence u(i) and output sequence $\tilde{y}(i)$,

$$\tilde{y}(i)+a_1\tilde{y}(i-1)+\ldots+a_n\tilde{y}(i-n) = b_0u(i-k)+b_1u(i-k-1)+\ldots+b_nu(i-k-n) \qquad (1)$$

where i is the integer time index, a_j and b_j are constant coefficients and k is the deadtime. The system is linear with respect to input-output behavior. This is equivalent to the transfer function equation

$$\tilde{Y}(z)=H(z)U(z) = \frac{B(z^{-1})}{1+A(z^{-1})}\,U(z) = \frac{(b_0+b_1z^{-1}+\ldots+b_nz^{-n})z^{-k}}{1+a_1z^{-1}+\ldots+a_nz^{-n}} \qquad (2)$$

OPTIMIZATION/PARAMETER ESTIMATION

The unconstrained minimization problem may be posed as

$$\min_{\underline{f}} Q = \varepsilon\{\sum_{i=1}^{N}(\tilde{y}_i-\overset{v}{y}_i)^2\} \qquad (3)$$

where $\underline{f}$ is the vector of the decision variables, i.e. model parameters and the deadtime,

$$\underline{f} = [k,\ b_0,\ \ldots,\ b_n,\ a_1,\ \ldots,\ a_n]^T \qquad (4)$$

and $\tilde{y}_i$ and $\overset{v}{y}_i$ are the model output and the noise corrupted process output, respectively. The necessary and sufficient conditions for $\underline{f}^*$ to be a local minimum are that the gradient evaluated at $\underline{f}^*$ be zero and that the Hessian matrix at $\underline{f}^*$ be positive definite.

Note that the objective function is nonlinear and non-quadratic in the model parameters. In this case, Powells' method requires the evalu-

ation of the first and second order derivatives of the objective function which, in turn, are based on the derivatives of the model output. The partial derivatives are obtained by differentializing (3)

$$\nabla Q = \frac{\partial Q}{\partial \underline{f}} = \varepsilon \{ \sum_{i=1}^{N} (\tilde{y}_i - \overset{v}{y}_i) \frac{\partial \tilde{y}_i}{\partial \underline{f}} \} \tag{5}$$

and

$$^2Q = \frac{\partial^2 Q}{\partial \underline{f} \partial \underline{f}^T} = \varepsilon \{ \sum_{i=1}^{N} \frac{\partial \tilde{y}_i}{\partial f^T} \cdot \frac{\partial \tilde{y}_i}{\partial \underline{f}} + \sum_{i=1}^{N} (\tilde{y}_i - \overset{v}{y}_i) \frac{\partial^2 \tilde{y}_i}{\partial \underline{f}^T \partial \underline{f}} \} . \tag{6}$$

The derivatives of the model output are obtained by formal differentiation of (2). This yields a recursive algorithm for computing the objective function and its partial derivatives: all are obtained as outputs of ARMA linear dynamical systems. (The derivatives with respect to different parameters in the same group are obtained simply by shifting). The details are similar to Astrom's maximum likelihood algorithm.

BIAS ANALYSIS

To show that this particular algorithm yields asymptotically unbiased estimates, consider the gradient of the objective function as expressed in equation (5),

$$\nabla Q = \varepsilon \{ \sum_{i=1}^{N} (\tilde{y}_i - \overset{v}{y}_i) \frac{\partial}{\partial \underline{f}} \tilde{y}_i - n_i \frac{\partial}{\partial \underline{f}} \tilde{y}_i \} \tag{7}$$

where $\overset{v}{y}_i = y_i + n_i$ and y_i is the true output and n_i is the output observation noise.

Note that the derivative of $\tilde{y}_i$ with respect to $\underline{f}$ depends on the input u_i but not on the noise n_i. If the current value of the noise, is independent of the current and previous model input values and if the noise has zero mean, then the gradient is zero if $\tilde{y}_i = y_i$ for any i, that is if $\tilde{H}(z) = H(z)$. This condition is easy to satisfy in open-loop; in closed-loop it requires that the noise be white and that there be at least one unit of delay in the controller. Simulation studies performed both in open-loop and in closed-loop with inferential controller have fully supported this finding.

SIMULATION

The class of controlled input is restricted to white noise signals which are independent of the disturbances acting on the process. The output noise sequence is such that $\varepsilon\{n_i\} = 0$ and $\text{cov}(n_i) = \varepsilon\{n_i n_2^T\} = \sigma^2 I$, I being the identity matrix. Table 1 shows the estimated parameters for a 2nd order system for different noise levels.

MOVING HORIZON STABILIZATION OF RANDOM PARAMETER SYSTEMS

E. YAZ
Applied Mathematics Department
Research Institute for Basic Sciences
P.O.Box 74 Gebze-Kocaeli, Turkey

ABSTRACT

We study the stabilizing property of a class of moving horizon control laws. First, the stabilization results for deterministic systems and a general quadratic criterion are extended to systems with singular state transition matrices and horizon lengths less than the controllability indices of the systems. Second, when the state transition matrix contains stochastic components, stochastic stabilization with probability one and in mean square are proved.

1. DETERMINISTIC CASE

Consider the linear, discrete-time system

$$x_{k+1} = Ax_k + Bu_k \tag{1}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ and A,B are constant matrices of appropriate dimensions. We adopt a constant feedback control strategy

$$u_k = Cx_k \tag{2}$$

which minimizes the moving quadratic cost functional

$$J = x_{k+N}^T Fx_{k+N} + \sum_{i=k}^{k+N-1} (x_i^T Qx_i + u_i^T Ru_i) \tag{3}$$

where $F^T=F>0$, $Q^T=Q>0$, and $R^T=R>0$. The required feedback gain is found to be

$$C = -(R+B^T P_{N-1} B)^{-1} B^T P_{N-1} A \tag{4}$$

where P_{N-1} is found by iterating the difference equation

$$P_{i+1} = Q + A^T (P_i^{-1} + BR^{-1}B^T)^{-1} A \tag{5}$$

starting with $P_o=F$. For the derivations of this control strategy and its properties useful to the designer, see Yaz [1]. We introduce two lemmata which will aid us in our analysis:

LEMMA. 1.1. All P_i, i=1,2,.... defined by (5) are positive definite.

PROOF. Simply follows from (5), noticing that F>0 and Q>0.

LEMMA 1.2. If the weighting matrices in the cost functional (3) are so chosen

that

$$F \geq Q+A^T(F^{-1}+BR^{-1}B^T)^{-1}A \tag{6}$$

then $P_{i+1} \leq P_i$ for all i=1,2,....

PROOF. The proof is by induction. From (6), $P_1 \leq P_o$. Suppose that $P_i \leq P_{i-1}$, it is easy to show that $P_{i+1} \leq P_i$.

Now we give the main stability result for the deterministic case:

THEOREM 1.1. If (6) holds, then all P_N, N>1, when used in (4) renders (1) asymptotically stable.

PROOF. We choose the Lyapunov function $V_k=x_k^T(P_{N-1}+P_{N-1}BR^{-1}B^TP_{N-1})x_k$. The closed loop matrix after application of (4) is found as

$$A_c = P_{N-1}^{-1}(P_{N-1}^{-1}+BR^{-1}B^T)^{-1}A$$

Forming the difference $\Delta V_k \triangleq V_{k+1}-V_k$, and after some manipulations, we obtain

$$\Delta V_k = -x_k^T(P_{N-1}-P_N+Q+P_{N-1}BR^{-1}B^TP_{N-1})x_k$$

By Lemma 1.2. $P_{N-1}-P_N \geq 0$ and since Q>0, it follows that $\Delta V_k < 0$, for all x_k.

The above stabilization problem was posed by Kwon and Pearson [2] and solved by them for the special case of F=∞I, A is an invertible matrix, and $N>N_c$=the controllability index of the system. The extension to F matrices with finite entries was done by Yaz [1].

2. STOCHASTIC CASE

Consider the system

$$x_{k+1}=(A+A_k)x_k+Bu_k \tag{7}$$

This time, A_o, A_1,.... are purely random matrices which are independent of each other and x_o. Their means $E\{A_k\}=0$ and $E\{a_{j\ell}^{(k)} \; a_{pq}^{(k)}\}$ are finite and constant in time, where $a_{j\ell}^{k}$, $1 \leq j, \ell \leq n$, are the entries of A_k. The feedback gain C_s which minimizes $E\{J\}$ where J has been defined by (3) is

$$C_s = -(R+B^TP_{N-1}^sB)^{-1}B^TP_{N-1}^sA \tag{8}$$

P_{N-1}^s is found from

$$P_{i+1}^s=Q+E\{A_o^TP_i^sA_o\}+A^T(P_i^{s\,-1}+BR^{-1}B^T)A \tag{9}$$

starting with $P_o^s=F$. A similar line of reasoning to deterministic case is used in deriving (8) and (9). Similar lemmata are also valid.

LEMMA 2.1. Under the same conditions on the weighting matrices, all P_i^s, $i\geq 1$, defined by (9) are positive definite.

PROOF. The proof is the same as in Lemma 1.1 except that, in this case, we should make use of the fact that $A(X)\triangleq E\{A_k^T X A_k\}$ is a positive operator, that is, for any $X\geq 0$, $A(X)\geq 0$. For this fact, see De Köning [3].

LEMMA 2.2. If we choose our weighting matrices such that

$$F>Q+E\{A_o^T F A_o\}+A^T(F^{-1}+BR^{-1}B^T)^{-1}A \tag{10}$$

then $P_{i+1}^s \leq P_i^s$ for all $i\geq 1$.

PROOF. Also by induction as in Lemma 1.2. The only difference is that now we use the fact that A is a monotonic transformation, that is, for $X\geq Y\geq 0$, it is true that $E\{A_k^T X A_k\}\geq E\{A_k^T Y A_k\}$. See de Köning [3] for the monotonicity property. Now, using these two lemmata, we can prove our stochastic stability result:

THEOREM 2.1. If we choose the weighting matrices such that (10) holds, then the application of (8) in (7) for any P_{N-1}^s, $N>1$, renders the closed loop system, both asymptotically stable with probability one and exponentially stable in mean square.

PROOF. Choosing the Lyapunov function $V_k=x_k^T P_{N-1}^s (P_{N-1}^{s\,-1}+BR^{-1}B^T)P_{N-1}^s x_k$, we obtain after a little manipulation,

$$E\{\Delta V_k\}\triangleq E\{V_{k+1}|x_k,\ldots,x_o\}-V_k=-x_k^T(Q+P_{N-1}^s BR^{-1}B^T P_{N-1}^s-E\{A_k^T P_{N-1}BR^{-1}B^T P_{N-1}A_k\}$$

$$+P_{N-1}^s-P_N^s)x_k \tag{11}$$

where the above expected value is conditional on x_i, $0\leq i\leq k$. $P_{N-1}^s-P_N^s\geq 0$ by lemma 2.2. A is a linear and monotonic transformation (de Köning [3]). Hence, A is a stable transformation (Zabczyk [4]) by (10), since $F\geq A(F)+Q$ with $F>0$ and $Q>0$, which means that the spectrum of A is contained in the unit circle. So I being the identity transformation, $(I-A)(X)\geq 0$ for any $X\geq 0$, and the corresponding terms in the brackets in (11) add up to a nonnegative definite matrix. Since $Q>0$, we have $E\{\Delta V_k\}<0$ for all x_k. So the system (7) with control (8) applied to it is asymptotically stable with probability one (Kushner [5]). For the above choise of V_k, we can obviously find positive real numbers α and $\beta>1$ such that

$$\alpha||x_k||_E^2 \leq V_k \leq \beta\ ||x_k||_E^2 \tag{12}$$

where $||x_k||_E$ is the Euclidean norm of x_k. Since $E\{\Delta V_k\}<0$, we can also find

an $\varepsilon<1$ such that

$$E\{V_{k+1}\} - E\{V_k\} < -\varepsilon E\{x_k^T x_k\} \tag{13}$$

Combining (12) and (13)

$$E\{V_{k+1}\} \leq (1-\frac{\varepsilon}{\beta})E\{V_k\} \tag{14}$$

Using these,

$$E\{||x_k||^2\} \leq \frac{1}{\varepsilon} E\{V_k\} \leq \frac{1}{\varepsilon} (1-\frac{\varepsilon}{\beta})^k E\{V_o\} \leq \frac{\beta}{\varepsilon} (1-\frac{\varepsilon}{\beta})^k E\{||x_o||_E^2\} \tag{15}$$

which means that the system is exponentially stable in mean square (Morozan [6]).

3. CONCLUSIONS

The results proved here relaxes the conditions of earlier stabilization results for deterministic systems utilizing moving performance criteria to be minimized. In the random parameter case, a constant feedback law is used which is different, for the first time, from the usual steady state linear quadratic optimal control. Even though, in this work, we have covered the case where the randomness enters only in the form of state dependent noise, we are currently searching for the possibility of including a control dependent noise term in our analysis.

REFERENCES

[1] Yaz, E., "A General Suboptimal Approach to the Control and Estimation of Discrete Time Systems" Int.J.of Systems Science (to appear).

[2] Kwon, W.H., and Pearson, A.E., "On Feedback Stabilization of Time-Varying Discrete Linear Systems", IEEE Trans.Auto.Contr., Vol.AC-23,No.3,June,1978.

[3] De Köning, W.L., "Infinite Horizon Optimal Control of Linear Discrete Time Systems with Stochastic Parameters", Automatica, Vol.18, p.443,1982.

[4] Zabczyk, J., "Stochastic Control of Discrete-Time Systems", Control Theory and Topics in Funct.Analy., IAEA, Vienna, Vol.3, p.187, 1976.

[5] Morozan, T., "On the Stability of Stochastic Discrete Systems", Control Theory and Topics in Funct.Analy., IAEA, Vienna, Vol.3, p.225,1976.

TIME OPTIMAL SEMI-MARKOV DECISION PROCESSES UNDER A CONSTRAINT

Frederick J. Beutler and Keith W. Ross
The University of Michigan, Ann Arbor, Michigan

ABSRACT

Considered are optimal causal policies maximizing the time-average reward over a semi-markov decision process (SMDP), subject to a hard constraint on a time-average cost. Rewards and costs depend on the state and action, and contain running as well as switching components. It is supposed that the state space of the SMDP is finite, and the action space compact metric. Under an accessibility hypothesis, several notions of time average are equivalent. Further, it is possible to utilize a Lagrange multiplier formulation involving a dynamic programming equation (DPE), to relate the constrained optimization to an unconsrained optimization parametrized by the multiplier. This approach leads to a proof for the existence of a semi-simple optimal constrained policy. Under a semi-simple policy, there is at most one state for which the action is randomized between two possibilities; at all other states, an action is uniquely chosen for each state.

I. INTRODUCTION

In an earlier work, the authors analyzed discrete-time optimal policies for controlled Markov chains [2], the objective being to maximize the average reward by a choice of policy that also met a specified average cost constraint. In this paper, we extend our results to a class of continuous time processes, namely semi-markov decision processes (hereafter designated SMDP).

We assume a finite state space, a compact action space, continuity of probabilities and rewards respective to the actions, plus an accessibility condition. Under reasonable hypotheses, this structure leads to the existence of a constrained optimal policy. The optimal policy is always stationary. It is either non-randomized stationary (i.e, simple), or acts like a simple policy except at one state; whenever the system is in that state, one chooses independently between two actions by the toss of a (biased) coin.

Proof of the above occupies the substance of Section II. It is shown there that various versions of the long term averages are equivalent under stationary policies. With the aid of accessibility and continuity assumptions, one then demonstrates the existence of an optimal constrained policy which is not only stationary, but is semi-simple in the sense of the preceding paragraph.

Basic to all our considerations is the (continuous parameter) semi-markov process $\{Y(t)\}$ defined on a finite state space $S=\{0,1,\ldots,N\}$, and assuming successive random values X_n during sojourn periods τ_n, $n=0,1,\ldots$ In other words, if we take $T_0=0$ and

$$T_n \triangleq \sum_{k=0}^{n-1} \tau_k \quad ,$$

then

$$Y(t)=X_n \quad \text{for} \quad T_n \le t < T_{n+1} \ .$$

Consequently $\{Y(t)\}$ is equivalently specified in terms of a Markov renewal process $\{X_n, T_n\}$, as described in [3], Section 10.5 and elsewhere. In accordance with the standard conventions for the Markov renewal process, its transition probabilities satisfy

$$P[X_{n+1}=x_{n+1}, \tau_n \le t \mid X_n, \tau_{n-1}, \cdots, \tau_0, X_0] = P[X_{n+1}=x_{n+1}, \tau_n \le t \mid X_n] \ . \quad (1.1)$$

For the most part, one may deal interchangeably with the Markov renewal process and the semi-markov process.

It may not be immediately obvious why SMDP, rather than controlled Markov processes, are of primary interest. After all, Markov processes are often viewed as the most suitable model for the study of communication networks[7], computer operating systems [8],[11], and queueing systems [6]. Indeed, the control of Markov processes with bounded transition rates has been treated by a device (see [3], Section 8.4) that leads back to the simpler case of discrete-time optimization (as in [5][10][12]).

There then arises the following question: if Markov jump processes represent a broad and interesting family of applications, why is it necessary to consider the more complex theory of SMDP? The answer is as follows: when the control permits the resulting process to retain the Markov property, reduction to discrete-time optimization is possible, as indicated in the preceding paragraph. But constrained optimization usually calls for a randomized action, which means that the conditional sojourn time distribution is a mixture of exponentials. This is inconsistent with the required exponential form of this distribution for a Markov process. In other words, a process that is Markov with non-randomized control often loses the Markov property when randomized controls are applied.

We begin by describing the *action* (or *control*), which is chosen at each jump instant T_k, to determine the joint distribution of τ_k and X_{k+1}, given X_k. An action a belongs to the *action space* α, which is a compact metric space equipped with a σ-algebra generated by its open sets. To exhibit how the action influences the probabilistic law of the SMDP, first let

$$H_n = (X_0, A_0, \tau_0, \cdots, X_n, A_n, \tau_n) \quad (1.2)$$

be a *history* of $\{(X_k, \tau_k)\}$; such a history evidently consists of the entire past of the process (up to epoch n), including also the actions $A_k \in \alpha$ taken at each epoch up to n. An action then parametrizes the transition probability in accordance with

$$P[X_{n+1}=y, \tau_n \le t \mid H_{n-1}, X_n=x, A_n=a] = P[X_{n+1}=y, \tau_n \le t \mid X_n=x, A_n=a]; \quad (1.3)$$

in other words, the controlled transition probability depends on the past only through the applied action. The right side of (1.3) is called the *law of motion*, which can be expressed in the form of the *controlled semi-markov kernel*

$$Q(x,y,t;a) \triangleq P[X_{n+1}=y, \tau_n \le t \mid X_n=x, A_n=a] \ . \quad (1.4)$$

It is required that this kernel is a conditional distribution on t in the wide sense (see [4], p. 29 ff.).

The choice of actions is determined by a *policy* or *strategy*. In general, the policy $\mathbf{u}$ in the *policy space* (designated U) can be described as $\mathbf{u}=\{u_0, u_1, \ldots\}$, where u_k is applied at epoch k. Specifically,

$u_k(\cdot \mid X_k = x, H_{k-1})$ is a conditional probability measure in the wide sense over $\mathcal{A}$; this measure is applied to the kernel to yield the conditional joint probability for X_{k+1} and τ_k, given X_k and H_k.

There are interesting subspaces of U that are not only easier to analyze, but represent policies easier to implement. In particular, a *stationary policy* $\mathbf{f}$ in space F is constituted by a probability measure over $\mathcal{A}$ that is conditioned only on the preceding state, and does not depend on the epoch index. More explicitly, we write $m_{\mathbf{f}}(\cdot \mid x)$ to represent the stationary policy $\mathbf{f}$; for any measurable set $B \subset \mathcal{A}$

$$m_{\mathbf{f}}(B \mid x) = P_{\mathbf{f}}[A_k \in B \mid X_k = x] = u_k(B \mid X_k = x, H_{k-1}) \tag{1.5}$$

The kernel under this policy $\mathbf{f}$ is then

$$Q_{\mathbf{f}}(x, y, t) = \int_{\mathcal{A}} Q(x, y, t; a)\, m_{\mathbf{f}}(da \mid x) \ . \tag{1.6}$$

A still more restricted class G is that of *simple* or *non-randomized stationary* policies. These are obtained by specializing the measure $m_{\mathbf{f}}$ to consist of a single atom. In that case, G can be characterized by a simple mapping, namely $\mathbf{g}: S \to \mathcal{A}$; hence, $g(x)$ acquires meaning as an element of $\mathcal{A}$, and $\mathbf{g}$ is viewed as a deterministic vector. For any $\mathbf{g} \in G$, (1.6) becomes $Q_{\mathbf{g}}(x, y, t) = Q(x, y, t; g(x))$.

In our earlier paper [2] we introduced *mixed* policies, whose space is indicated by ${}_mF$. A mixed policy is merely a stationary policy with atoms of mass q and $1-q$ for each $x \in S$. Such a policy may be written symbolically for convenience $\mathbf{f}_q = q\,\mathbf{g}_1 + (1-q)\mathbf{g}_2$, with $q \in [0,1]$. The meaning attached to such a $\mathbf{f}_q$ is that a randomized choice between the two policies $\mathbf{g}_1$ and $\mathbf{g}_2$ is to be made at each decision point. If the state is x at that instant, action $g_1(x)$ is invoked with probability q, and $g_2(x)$ with probability $1-q$.

Finally, we define the class of *semi-simple* policies, which is designated ${}_sF$. We say $\mathbf{f} \in {}_sF$ if $\mathbf{f} \in F$ is characterized by a measure $m_{\mathbf{f}}(\cdot \mid x)$ possessing the following property: there is a state $x_0 \in S$ for which $m_{\mathbf{f}}(\cdot \mid x_0$ is described by either one or two atoms; for every other state, $m_{\mathbf{f}}(\cdot \mid x)$ consists of exactly one atom. In other words, there may be one state for which randomization between two actions is required, while the policy is non-random for every other state.

The system earns a *reward* that accumulates throughout time. The reward consists of two components; one takes on an increment with every jump, and the other is a "running reward" whose rate is specified. To reflect this, we write

$$C(X_k, A_k) = c_0(X_k, A_k) + \tau_k c(X_k, A_k) \tag{1.7}$$

to denote the incremental reward earned during the sojourn interval $[T_k, T_{k+1})$ when the state is X_k and the action A_k is applied. In terms of the above, we seek to maximize the *average reward* defined by

$$R_x(\mathbf{u}) \triangleq \liminf_n \left\{ \frac{E_{\mathbf{u}}^x[\sum_{k=0}^{n-1} C(X_k, A_k)]}{E_{\mathbf{u}}^x[\sum_{k=0}^{n-1} \tau_k]} \right\}, \tag{1.8}$$

in which $E_{\mathbf{u}}^x$ is the conditional expectation when the probability measure is

determined by the policy $\mathbf{u}$, and the conditioning event is $\{X_0=x\}$. Ordinarily, it is desired to find the $\mathbf{u}\in U$ that yields the supremum of the average reward (1.8) over $\mathbf{u}\in U$ for all $x\in S$, if such exists; however, we have yet to consider the supremum only over those policies that satisfy a constraint we are about to specify. To this end, let the system incur a *cost* called $D(\cdot,\cdot)$ with definition and notation entirely analogous to those of the reward. The *average cost* is given by

$$K_x(\mathbf{u}) \triangleq \limsup_n \left\{ \frac{E_{\mathbf{u}}^x[\sum_{k=0}^{n-1} D(X_k,A_k)]}{E_{\mathbf{u}}^x[\sum_{k=0}^{n-1} \tau_k]} \right\}. \tag{1.9}$$

We postpone for the time being any discussion on the measurability of $C(x,\cdot)$ and $D(x,\cdot)$.

Our *constraint* is on the average cost, in the sense that we require

$$K_x(\mathbf{u}) \le \alpha \tag{1.10}$$

for all x. Then, if U_0 is the subspace of U on which the constraint (1.10) is satisfied, we shall discuss the attainment of

$$R_x = \sup_{\mathbf{u}\in U_0} R_x(\mathbf{u}) \tag{1.11}$$

Any policy $\mathbf{u}$ that attains R_x for each x while simultaneously satisfying the constraint (1.10) is termed a *constrained optimal policy*, or more simply, an *optimal policy*.

II. THE CONSTRAINED OPTIMAL POLICY

It is clear that U, the family of all causal policies, encompasses the broadest class over which one might wish to optimize. Nevertheless, F, the class of stationary policies, will be of principal concern in what follows. This is justified on two counts. First, it is shown that there exists a constrained optimal policy belonging to F; although there is the possibility of equally good policies in $U-F$, little is known about their existence or nature. Second, policies in F, in conjunction with certain reasonable assumptions (needed for the optimization theory anyway), lead to alternative expressions for the average reward that are more appealing than (1.8).

We note also at this juncture that a policy $\mathbf{f}\in F$ maintains the semi-markov property (1.1), taken of course with respect to the probability measure implied by this $\mathbf{f}$. To show that the semi-markov equation is satisfied, one combines (1.3) with the fact that $m_{\mathbf{f}}(\cdot\,|X_k,H_{k-1})=m_{\mathbf{f}}(\cdot\,|X_k)$ by the definition of a stationary policy. With the semi-markov structure in place, it becomes possible to investigate the limit properties of the SMDP, and to draw the consequent inferences on suitable expressions for the reward.

The first set of assumptions assure the positive recurrence of the SMDP. We begin with some characteristics of the sojourn times τ_n. Let us define

$$\tau(x,a) \triangleq \int_0^\infty [1-\sum_{y\in S} Q(x,y,t;a)]\,dt \tag{2.1}$$

so that clearly

$$\tau(x,a) = E[\tau_n \mid X_n = x, A_n = a] \quad . \tag{2.2}$$

We shall require

Hypothesis 2.1: $\tau(x,\cdot)$ is continuous on $\mathcal{A}$, and

$$\sum_{y \in S} Q(x,y,0;a) < 1$$

for each $x,y \in S$ and each $a \in \mathcal{A}$.

It follows from the second part of Hypothesis 2.1 that $\tau(x,a) > 0$. Then the assumed continuity, together with the compactness of $\mathcal{A}$ assures a positive lower bound for $\tau(x,a)$, and the same properties also demand a finite upper bound. Therefore,

$$0 < m \leq \tau(x,a) \leq M < \infty \quad . \tag{2.3}$$

The next assumption makes certain that S contains only one positive recurrent class, which to a certain extent does not depend on the choice of policy. Other, less restrictive hypotheses are possible, but these can complicate the results without enhancing insight into the system behavior.

Hypothesis 2.2: For each simple policy $\mathbf{g}$, there is a state (take 0 for definiteness) which is accessible from each $x \in S$ for the Markov chain with transition probabilities

$$P_{xy}(\mathbf{g}) \triangleq Q(x,y,\infty;g(x)) \tag{2.4}$$

In view of these two hypotheses, the expected hitting time for the state 0 is finite, and in fact, the successive times at which this state is entered constitute regeneration points for $Y(t)$. We formalize this idea by

Lemma 2.3: Let T be the hitting time for the state 0. If Hypotheses 2.1 and 2.2 are valid,

$$\sup_{\mathbf{f} \in F} E_{\mathbf{f}}^{x}(T) < \infty \quad , \tag{2.5}$$

and $\{Y(t)\}$ is a regenerative process relative to the successive hitting times for 0.

Proof: Let N be the corresponding hitting time for the discrete-time process $\{X_n\}$. As is shown in [2], Theorem 2.6,

$$\sup_{\mathbf{f} \in F} E_{\mathbf{f}}^{x}(N) < \infty \tag{2.6}$$

and we also have

$$T = \sum_{0}^{N-1} \tau_k \quad .$$

But by (2.3), $E_{\mathbf{f}}^{x}(\tau_k) \leq M$ for all $x \in S$ and $\mathbf{f} \in F$, so by an argument similar to that used in the derivation of Wald's lemma, (2.5) is readily verified. The regenerative property is implied by (2.5); see [9], Theorem 7.5.

It is convenient (though not mandatory) that the Markov chain specified by the transition matrix $\mathbf{P}(\mathbf{f})$ be aperiodic, so that we are able to claim that the unique probability row vector $\nu_{\mathbf{f}}$ satisfying

$$\nu_{\mathbf{f}} = \nu_{\mathbf{f}} \mathbf{P}(\mathbf{f}) \quad ; \tag{2.7}$$

has the limit property $P_{\mathbf{f}}^{x}[X_n = y] \to \nu_{\mathbf{f}}(y)$. We further assume that the sojourn times τ_k are non-arithmetic (again to avoid superficial complexities), so that $\lim_t P_{\mathbf{f}}^{x}[Y(t) = y]$ exists, and is furnished by

$$\pi_{\mathbf{f}}(y) = \frac{\nu_{\mathbf{f}}(y)\tau_{\mathbf{f}}(y)}{\sum_{x}\nu_{\mathbf{f}}(x)\tau_{\mathbf{f}}(x)} \tag{2.8}$$

in which

$$\tau_{\mathbf{f}}(x) \triangleq \int_{\mathcal{A}} \tau(x,a) m_{\mathbf{f}}(da \mid x) \ . \tag{2.9}$$

In addition to conditions related to accessibility, we also need a continuity assumption, namely

Hypothesis 2.3: The functions $c_0(x,\cdot), c(x,\cdot), d_0(x,\cdot)$ and $d(x,\cdot)$ are all continuous on $\mathcal{A}$.

By virtue of Hypothesis 2.3 and the compactness of $\mathcal{A}$ these functions are all bounded, and we may therefore suppose them to be non-negative. The continuity will, of course, settle all measurability questions regarding these functions.

With the above hypotheses and facts, one can express the average reward in any one of several ways, some of which are more intuitively suggestive than (1.8).

Theorem 2.4: For policies in F, and any X_0, the average reward (1.8) can be written as

$$R(\mathbf{f}) = \frac{\sum_{x}\nu_{\mathbf{f}}(x) C_{\mathbf{f}}(x)}{\sum_{x}\nu_{\mathbf{f}}(x)\tau_{\mathbf{f}}(x)} \tag{2.10}$$

where $\tau_{\mathbf{f}}(x)$ is as previously defined, and

$$C_{\mathbf{f}}(x) \triangleq \int_{\mathcal{A}} \bar{C}(x,a)\, m_{\mathbf{f}}(da \mid x) \ , \tag{2.11}$$

with

$$\bar{C}(x,a) \triangleq c_0(x,a) + \tau(x,a) c(x,a) \ . \tag{2.12}$$

An alternative expression for the average reward is

$$R(\mathbf{f}) = \lim_{n} \frac{\sum_{0}^{n-1} c_0(X_k, A_k)}{\sum_{0}^{n-1} \tau_k} + \lim_{t} t^{-1} \int_{0}^{t} c\,(Y(\xi), A(\xi))\, d\xi \ , \tag{2.13}$$

almost surely. Here $A(t) = A_k$ for $t \in [T_k, T_{k+1})$. In case

$$c_0 \equiv 0 \quad \text{and} \quad C_{\mathbf{f}}(x) = \tau_{\mathbf{f}}(x) c_{\mathbf{f}}(x) \ , \tag{2.14}$$

with $c_{\mathbf{f}}(x)$ defined by

$$c_{\mathbf{f}}(x) \triangleq \int_{\mathcal{A}} c\,(x,a) m_{\mathbf{f}}(da \mid x) \ , \tag{2.15}$$

(2.10) becomes

$$R(\mathbf{f}) = \sum_{x} \pi_{\mathbf{f}}(x) c_{\mathbf{f}}(x) \ ; \tag{2.16}$$

equivalently

$$R(\mathbf{f}) = \lim_{t\to\infty} t^{-1} \int_0^t c_{\mathbf{f}}(Y(\xi))\, d\xi \tag{2.17}$$

almost surely.

Proof: For the term $E_{\mathbf{f}}^{x}[\tau_k c(X_k, A_k)]$ in (1.8) calculate by the chain rule

$$\begin{aligned} E_{\mathbf{f}}^{x}[\tau_k c(X_k, A_k)] &= E_{\mathbf{f}}^{x}\{c(X_k, A_k) E_{\mathbf{f}}[\tau_k \mid X_k, A_k, H_{k-1}]\} \\ &= E_{\mathbf{f}}^{x} \int_a c(X_k, a)\tau(X_k, a) m_{\mathbf{f}}(da \mid X_k, H_{k-1}) \ , \end{aligned} \tag{2.18}$$

where we have also used the semi-markov property and (2.2). Now apply (1.5), obtaining from (2.18) the expression

$$\sum_y \{\int_a c(y, a)\tau(y, a) m_{\mathbf{f}}(da \mid y)\} P_{\mathbf{f}}^{x}[X_k = y \mid H_{k-1}] \ . \tag{2.19}$$

But $P_{\mathbf{f}}^{x}[X_k = y \mid H_{k-1}] = P_{\mathbf{f}}^{x}(X_k = y)$, again by the semi-markov property under $\mathbf{f}$. Finally, since the sum on the right of the above equation is finite, and the integrand bounded, the equality

$$\lim_{k\to\infty} P_{\mathbf{f}}^{x}(X_k = y) = \nu_{\mathbf{f}}(y) \ ,$$

completes the argument for one term of (2.10). The remaining terms in (2.10) are a specialization of what we have just done, and may be obtained in one case by taking $c \equiv 1$, and in the other by taking $\tau \equiv 1$ and $c \equiv c_0$.

The claim (2.13) merely represents a restatement on the equivalence of average rewards in [9], page 159. As for (2.16), one need only substitute (2.14) and (2.8) in (2.10). Then (2.17) follows from Theorems 5.10 and 5.16 in [9].

Corollary 2.5: For each policy $\mathbf{g} \in G$, there is a Markov chain $\{\hat{X}_k\}$ having the same recurrence properties as $\{X_k\}$, and defined on the same state space S. For this new Markov chain with transition matrix $\hat{\mathbf{P}}(\mathbf{g})$, the associated (unique) probability vector vector $\pi_{\mathbf{g}}$ given by (2.8). Moreover, when $c_0 \equiv 0$, $R(\mathbf{g})$ is furnished by

$$R(\mathbf{g}) = \lim_{n\to\infty} n^{-1} \hat{E}_{\mathbf{g}}[\sum_0^{n-1} c(\hat{X}_k, g(\hat{X}_k))] \tag{2.20}$$

Proof of Corollary: Define the matrix

$$\hat{P}_{xy}(a) = \begin{cases} \dfrac{m}{\tau(x,a)} P_{xy}(a) & x \neq y \\ 1 - \dfrac{m}{\tau(x,a)}[1 - P_{xx}(a)] & x = y \end{cases} \ , \tag{2.21}$$

where m is as in (2.3). It is easily checked that $\hat{\mathbf{P}}(\mathbf{g})$ is a transition matrix each of whose off-diagonal elements are zero only when the same is true of the corresponding element of $\mathbf{P}(\mathbf{g})$. Also, each diagonal element of $\hat{\mathbf{P}}(\mathbf{g})$ is positive, except perhaps when the same diagonal element of $\mathbf{P}(\mathbf{g})$ is zero. Thus, there is a new Markov chain as claimed, with only one aperiodic recurrent class that must include the state 0. The stationary probability vector is then unique, and so we need only verify that the $\pi_{\mathbf{g}}$ of (2.8) solves

$$\pi_g = \pi_g \hat{P}(g) \quad . \tag{2.22}$$

To prove this, combine (2.7) and (2.8), noting that for a simple policy $\tau_g(y)=\tau(y,g(y))$ and that $\hat{P}_{xy}(g)=\hat{P}_{xy}(g(x))$.

The validity of (2.20) still needs to be shown. For a policy in G, $c_g(x)=c(x,g(x))$ in (2.16). It remains to apply a standard result relating expectations and time averages for additive functionals on Markov chains; see [3], Theorem 2.26 of Chapter 6.

Let us now introduce the Lagrangian constraint, which appears in terms of the reward $C(\cdot,\cdot)$ as well as the cost $D(\cdot,\cdot)$. In terms of a constraint parameter ω, we consider the unconstrained optimization for the *parametrized reward*

$$B^{\omega}(x,a) \stackrel{\Delta}{=} C(x,a)-\omega D(x,a) \quad . \tag{2.23}$$

The parametrized reward has all the properties of the reward (i.e., boundedness, continuity), except that it need not be non-negative. There is now a *parametrized average reward* $J_x^{\omega}(\mathbf{u})$ completely analogous to (1.8), to which Theorem 2.4 is immediately applicable.

Central to the derivation of the optimal parametrized average reward is *Theorem 2.6:* Suppose for fixed ω there exists a scalar s^{ω} and a bounded vector $\mathbf{h}^{\omega}$ such that the DPE

$$h^{\omega}(x) = \sup_{a\in A}[B^{\omega}(x,a)+\sum_{y\in S} P_{xy}(a)h^{\omega}(y)-s^{\omega}\tau(x,a)] \tag{2.24}$$

is satisfied for each $x\in S$. Then any policy $\mathbf{g}^{\omega}\in G$ specified by

$$g^{\omega}(x) = \arg\sup_{a\in A}[B^{\omega}(x,a)+\sum_{y\in S} P_{xy}(a)h^{\omega}(y)-s^{\omega}\tau(x,a)] \tag{2.25}$$

attains

$$J^{\omega} = \sup_{\mathbf{u}\in U} J_x^{\omega}(\mathbf{u}) \tag{2.26}$$

for all $x\in S$. Moreover, s^{ω} in (2.24) satisfies $s^{\omega}=J^{\omega}$.
Proof: See [9], Theorem 7.6.

We need to show that solution(s) indeed exist to (2.24). From that point on, the analysis in [2] will lead to the conclusion that there is an optimal constrained policy belonging to ${}_sF$. As for the first part, we claim
Theorem 2.7: A solution to (2.24) exists for each $\omega\geq 0$.
Proof: For fixed ω we may omit that parameter for convenience. Then take $H^s(x,a)=B(x,a)-s\tau(x,a)$, and consider

$$L^s+h^s(x) = \sup_{a\in A}[H^s(x,a)+\sum_{y\in S} P_{xy}(a)h^s(y)] \quad , \tag{2.27}$$

which has the same form as the constrained equations in Section III of [2], except that s (in place of ω) is now the relevant parameter, with the proviso that s could be negative. Moreover, the assumptions already made are precisely those of [2]. It follows at once that L^s has all the properties imputed to J^{λ} of the reference.

In particular, L^s in continuous and non-increasing in s, as shown in Lemma 3.1 of [2]. L^s is even strictly decreasing, with a derivative almost everywhere equal to

$$-\sum_x \pi_{g^s}(x)\tau(x,g^s(x)) \le -m < 0 \ ;$$

therefore, there is a unique s such that $L^s=0$. It follows that (2.24) is satisfied when this s appears in (2.27).

Remark: The above Theorem is valid for the unconstrained policy, which is obtained as a special case by setting $\omega=0$. Theorems such as the above are usually proved by studying the discounted cost problem, and then letting the discount factor approach zero [1],[9]. The method of Theorem 2.7 offers an alternate approach for the unconstrained SMDP.

We may now proceed as in Sections III and IV of [2]. However, we must first be assured that the problem is not rendered impossible by the non-existence of any policy capable of satisfying the average constraint (1.10). Accordingly, we require

Hypothesis 2.8: There exists a $\mathbf{g}\in G$ such that

$$K(\mathbf{g}) < \alpha \quad . \tag{2.28}$$

Remark: If $K(\mathbf{g})>\alpha$ for all $\mathbf{g}\in G$, the same is true for all $\mathbf{u}\in U$. Then no policy whatsoever is capable of meeting the constraint; see [2], Section V.

In view of the Hypothesis and the existence of the DPE solution, all of Section IV of [2] can be applied without essential change. Hence, we could claim the same conclusion found there, namely that there exists a constrained optimal policy in ${}_mF$. However, it would be more advantageous if randomization were needed for at most one state. We have recently been able to demonstrate a result to that effect, namely

Theorem 2.9: There exists a constrained optimal policy in ${}_sF$.

Proof: We recall the elements of the proof of Theorem 4.4 in [2]. Following the methodology of the referenced proof, one introduces the constraint parameter

$$\gamma \triangleq \inf_{\omega} \{\omega \,|\, K(\mathbf{g}^\omega)\le\alpha\} \quad . \tag{2.29}$$

where $\mathbf{g}^\omega$ is any simple policy that solves

$$h^\omega(x) = \sup_{a\in\mathcal{A}} [B^\omega(x,a)+\sum_{y\in S} P_{xy}(a)h^\omega(y)-J^\omega\tau(x,a)] \tag{2.30}$$

for each $x\in S$. That such a $\gamma<\infty$ exists follows from Hypothesis 2.8, using the argument of Lemma 3.3 in [2].

If there is a $\mathbf{g}^\gamma$ such that $K(\mathbf{g}^\gamma)=\alpha$ we are finished, according to Theorem 4.3 of [2]. Otherwise, we consider the policies G^γ in G that solve (2.30). As in Theorem 4.4 of [2], we use the compactness of $[0,\beta]\times G^\beta$ for any $\beta<\infty$ (see Theorem 3.5 of the referenced paper), together with the continuity of $K(\cdot)$ on G (Lemma 3.4 of the reference) to show that there exist $\underline{\mathbf{g}},\bar{\mathbf{g}}\in G^\gamma$ such that

$$K(\underline{\mathbf{g}})=\alpha_0<\alpha<\alpha^0=K(\bar{\mathbf{g}}) \quad . \tag{2.31}$$

In fact, G^γ contains not only $\underline{\mathbf{g}}$ and $\bar{\mathbf{g}}$, but also all elements $\mathbf{g}^{(k)}\in G$ of the form

$$g^{(k)}(x) = \begin{cases} \underline{g}(x) & x<k \\ \\ \bar{g}(x) & x\ge k \end{cases} \ ; \tag{2.32}$$

thus, $\mathbf{g}^{(0)}=\bar{\mathbf{g}}$, and for consistency we take $\mathbf{g}^{(n+1)}=\underline{\mathbf{g}}$. That all these $\mathbf{g}^{(k)}$ belong

to G^γ is true because $\underline{g}$ and $\bar{g}$ satisfy (2.30) separately for each coordinate $x \in S$.

Now take m as the first index such that $K(g^{(m)}) < \alpha$. Clearly, $0 < m \leq n+1$,

$$K(g^{(m)}) < \alpha < K(g^{(m-1)}) \quad , \tag{2.33}$$

and $g^{(m)}(x)$ differs from $g^{(m-1)}(x)$ only for $x = m-1$. To complete the proof, we proceed to randomize between $g^{(m)}$ and $g^{(m-1)}$ as in the last portion of the proof of Theorem 4.4 in [2]. If the randomization is symbolically written $f_q \triangleq q\, g^{(m)} + (1-q) g^{(m-1)}$, we find (just as in [2]) that $J^\gamma = J^\gamma(f_q)$, and that $K(f_q)$ is continuous in q. It then follows from (2.33) that for some $q \in (0,1)$ we have $K(f_q) = \alpha$; hence, by Theorem 4.3 of [2], this f_q is optimal. Since $f_q \in {}_sF$, the proof is complete.

Remark: In [2], where we treated the discrete time constrained optimization problem exclusively, we were able to show only that the optimal constrained policy belonged to ${}_mF$. The above proof strengthens this result of [2], in that we are now able to conclude that there exists an optimal constrained policy in ${}_sF$.

REFERENCES

[1] D.P. Bertsekas, *Dynamic Programming and Stochastic Control*, Academic Press, New York, 1976.

[2] F. Beutler, K. Ross, *Optimal Policies for Controlled Markov Chains with a Constraint*, March 1984, 18 pp., to appear in *J. Math. Anal. and Appl.*

[3] E. Cinlar, *Introducation to Stochastic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[4] J.L. Doob, *Stochastic Processes*,

[5] B. Hajek "Optimal control of two interacting service stations," *IEEE Trans. Automatic Control, AC-29* (1984), 491-499.

[6] F.P. Kelly, *Reversibility and Stochastic Networks*, Wiley, New York, 1979.

[7] L. Kleinrock, *Queueing Systems Volume II: Computer Applications*, Wiley, New York, 1976.

[8] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*, Addison-Wesley, Reading, MA, 1978.

[9] S.M. Ross, *Applied Probability Models with Optimization Applications* Holden Day, San Francisco, 1970.

[10] Z. Rosberg, P. Varaiya, J. Walrand, "Optimal control of service in tandem queues," *IEEE Trans. Automatic Control, AC-27* (1982), 600-610.

[11] C. Sauer, K. Chandi, *Computer Systems Performance Modeling*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[12] S. Stidham, Jr., "Optimal control of arrivals to queues and networks of queues," presented at the *21st IEEE Conference on Decision and Control*, Orlando, FL, December 1982.

ON THE INSTRUMENTAL VARIABLE METHOD

S. PUTHENPURA and N.K. SINHA,
Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada

ABSTRACT

A comparative study is made on the choice of instrumental variables for the identification of system parameters when the measurements are contaminated with noise. A powerful but computationally very simple method is suggested for constructing instrumental variables which works much better than the existing methods even when the signal to noise ratio is as low as zero decibels.

INTRODUCTION

The instrumental variable (IV) approach to identification of system parameters essentially employs two sets of data series. One is the actual input and output of the system to be identified and the other is the so-called instrumental variables. Several methods have been proposed for constructing instrumental variables [1] which will be briefly explained. Also we present another simple method which appears to work better in comparison with the existing methods. An example is also included to compare ALL the methods.

FORMULATION OF THE PROBLEM

Consider a linear discrete-time system described by the transfer function

$$H(z^{-1}) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_m z^{-m}}{1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}} \qquad (1)$$

Assume that this system is perturbed with a sequence of inputs $\{u(k)\}$ and the outputs $\{y(k)\}$ are measured for $k = 0, 1, 2, \ldots$, which are contaminated with measurement noise (generally coloured). Our aim is to estimate the parameters of (1) from $\{u(k)\}$ and $\{y(k)\}$. It is assumed that the orders m and n are known á priori. We can write

$$y(k) = \underline{\Phi}^T(k)\,\underline{\theta} + n(k)\,,\ k = 0, 1, 2, \ldots \qquad (2)$$

where $\underline{\Phi}^T(k) = [u(k), u(k-1), \ldots u(k-m), -y(k-1), -y(k-2), \ldots -y(k-n)]$;

$$\underline{\theta} = [a_0, a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n]^T \qquad (3)$$

and n(k) is the measurement noise. The basic idea of the IV technique is to construct

$$\underline{\Psi}^T(k) = [u(k), u(k-1), \ldots u(k-m), -h(k-1), -h(k-2), \ldots -h(k-n)] \qquad (4)$$

Such that $\underline{\Psi}^T(k)$ and n(k) are uncorrelated at the same time $\underline{\Psi}^T(k)$ and $\underline{\Phi}^T(k)$ are strongly correlated. The quantities $\{h(k)\}$ are known as the instrumented variables. One common practice is to pass $\{u(k)\}$ through a deterministic model (called the auxiliary model) and use its outputs as $\{h(k)\}$. The success of this method depends upon an intelligent guess on the parameters of the auxiliary model. The closer it is to the system model, the better the results. Another simple way is to take $\{h(k)\} = \{y(k-1)\}$. This method is computationally attractive but not as effective as other methods, since $\{h(k)\}$ generated in this fashion and the noise sequence $\{n(k)\}$ are not completely uncorrelated. A third alternative to the problem is to update the auxiliary model parameters with the latest estimate of system parameters. This works well compared to the other two, but the increase in computation makes it somewhat unattractive for on-line identification purposes.

THE NEW APPROACH

Let X_1 and X_2 be any two random variables with the same probability distribution. If we define $G_1 = X_1 - X_2$ and $G_2 = X_1 + X_2$, then,

$$\mathrm{Cov}(G_1, G_2) = E\{[X_1 - X_2][X_1 + X_2]\} = E[X_1^2 - X_2^2] = 0 \qquad (5)$$

This means that G_1 and G_2 are uncorrelated [2]. Now, the estimated value of the noise can be expressed as, $e(k) = y(k) - \underline{\Phi}^T(k)\,\underline{\theta}(k)$ where $\underline{\theta}(k)$ is the estimated value of the parameter vector $(\underline{\theta})$ at the kth iteration. Hence by choosing $h(k) = y(k) + \underline{\Phi}^T(k)\,\underline{\theta}(k)$, it follows from (5) that $\{e(k)\}$

and {h(k)} are uncorrelated. At the same time $\Phi(k)$ and $\psi(k)$ are strongly correlated. Hence {h(k)} obtained in this manner is a good choice for instrumental variables.

AN EXAMPLE

Consider the stiff system,

$$H(s) = \frac{s + 1}{(s + 0.1)(s + 10)} \tag{6}$$

Its impulse invariant z-domain model can be expressed as

$$H(z^{-1}) = \frac{1 - 0.965400\, z^{-1} + 0\, z^{-2}}{1 - 1.666328\, z^{-1} + 0.960789\, z^{-2}} \tag{7}$$

at a sampling interval of 0.04 seconds. The system was perturbed with a series of random inputs and the corresponding outputs were taken. Coloured noise was also added to these noise free outputs to get different SNR. The results are tabulated in Table 1. The parameters are given in the order a_0, a_1, a_2, b_1, b_2. The fixed auxiliary model used in method 1 was

$$\bar{H}(z) = \frac{z^2 + 0.5\, z}{z^2 - z + 0.5} \tag{8}$$

Table 1
Estimated Parameters of Model [7] Using Different IV

SNR in dB	Method 1 [using the fixed aux. model (8)]	Method 2 [{h(k)} = { y(k − 1)}]	Method 3 [Updating the aux. model]	The proposed Method
100	0.9999881	1.0000015	1.0000021	1.0000002
	−0.9664255	−0.9664019	−0.9664005	−0.9664003
	0.0000035	0.0000006	0.0000000	−0.0000001
	−1.6663208	−1.6663280	−1.6663274	−1.6663280
	0.9607727	0.9607893	0.9607832	0.9607890
0	−0.5353822	9.4855251		1.5752694
	−3.9164796	−1.4215574		−1.0856874
	2.2118256	−0.1646073	Failed to	−0.0228524
	−1.6942117	−0.1646073	Converge	−1.6142160
	0.9937714	0.9284818		0.8843486

CONCLUSIONS

A very simple but powerful method for constructing instrumental variables for identification is presented. Several simulated examples tried by the authors revealed the fast convergence of this scheme as well as the better performance compared to other equally simple methods when the signal-to-noise ratio was as low as zero decibel. Hence the method presented can be very useful for on-line identification in practical situations.

REFERENCES

1. Sinha, N.K. and Kuszta, B., "Modeling and identification of dynamic systems", Van Nostrand Reinhold Company, 1983.

2. Gibbons, J.D., "Mutually exclusive events, independence and zero correlation", The American Statistician, 22, No. 5, December 1968, pp. 31-32.

FINITE DIMENSIONAL COMPENSATOR DESIGN FOR LINEAR RETARDED SYSTEMS*

WU-SHENG LU, E. BRUCE LEE
Dept. of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455

STANISLAW H. Żak
School of Electrical Engineering,
Purdue University
West Lafayette, IN 47907

ABSTRACT

By taking advantage of canonical forms, explicit conditions are given such that an adequate compensator can be designed to meet requirements of stability for the delay systems. Relationships between realizability properties of the compensator such as causality and stability are also described.

I. INTRODUCTION

It has long been a tradition to seek explicit and checkable conditions for linear retarded systems to be arbitrarily coefficient-assignable by certain types of compensators [1] - [5]. Consider a single-input single output (SISO) delay system with mathematical model

$$\begin{aligned} \dot{x}(t) &= A(z)x(t) + b(z)u(t) \\ y(t) &= c(z)x(t) \end{aligned} \tag{1.1}$$

where $A(z)\in\mathcal{R}^{nxn}[z]$, $b(z)\in\mathcal{R}^{nx1}[z]$, $c(z)\in\mathcal{R}^{1xn}[z]$, $\mathcal{R}[z]$ is the ring of real polynomials in z, and z is a delay operator. Let $\mathbf{X} \triangleq \{(A(z), b(z)) \mid \det[A(z) \mid b(z)] \not\equiv 0\}$ where $[A \mid b] \triangleq [b, Ab, ..., A^{n-1}b]$ is the controllability matrix associated with the pair (A, b). Each element in $\mathbf{X}$ is associated with a $\mathcal{R}(z)$ - controllable system [7]. In [3] a characterization of the possible coefficients of $\det[sI - (A(z) + b(z)k(z))]$ is given whenever $(A(z),b(z))\in\mathbf{X}$ is fixed and $k(z)$ varies in $\mathcal{R}^{1xn}[z]$. Assuming $\det[sI - (A + bk)] = s^n + \sum_{i=1}^{n}\alpha_i(z)s^{i-1}$ and $\alpha(z) = [\alpha_1(z),...,\alpha_n(z)]$, it turns out [3] that the set of all possible $\alpha(z)$ forms a proper submodule in $\mathcal{R}^n[z]$ unless the considered pair (A,b) is $\mathcal{R}[z]$-controllable, namely $\det[A \mid b]$ = nonzero constant, which is obviously a nongeneric event for the single-input retarded systems.

In the celebrated survey [5], Sontag considered stabilization of delay-differential systems by viewing the polynomial ring $\mathcal{R}[z]$ as a subring of $\mathcal{R}_s[z] \equiv \{p(z)/q(z) \mid p,q\in\mathcal{R}[z] \text{ and } q \neq 0 \text{ for } |z| \leq 1\}$.

The work summarized here has shown that the set of possible coefficient vectors $\alpha(z)$ is (compared with [3]) considerably enlarged by admitting controllers of the

* Research supported in part by the National Science Foundation under Grant No. ECS 8217375, in part by a Ph.D. Dissertation Fellowship from the Graduate School of the University of Minnesota, and in part by the School of Electrical Engineering of the Purdue University.

form $u(t) = k(z)x(t) + v(t)$, where $v(t)$ is an external input and $k(z) = w^{-1}(z)N(z)$ where $w(z) \in \mathcal{R}[z]$, $N(z) \in \mathcal{R}^{1xn}[z]$ rather than the restriction to $k(z) \in \mathcal{R}^{1xn}[z]$. Actually, by taking advantage of canonical forms, explicit conditions are given under which an adequate compensator, as mentioned above, yields a desired coefficient vector $\alpha(z)$. The relationships between realizability properties such as causality and stability of the compensator and structural properties of the pair (A,b) are also described. Two examples illustrate the simplicity of the compensator design procedure.

II. MAIN RESULTS

Let $(A(z), b(z)) \in \mathbf{X}$ and write

$$[A(z)|\, b(z)]^{-1} = \begin{bmatrix} h_1(z) \\ h_2(z) \\ \vdots \\ h_n(z) \end{bmatrix} \quad \text{and} \quad T(z) = \begin{bmatrix} h_n \\ h_nA \\ \vdots \\ h_nA^{n-1} \end{bmatrix} \tag{2.1}$$

Then

$$\hat{A}(z) \equiv TAT^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & 0 & \cdots & 1 \\ -a_1(z) & \cdot & \cdot & \cdots & -a_n(z) \end{bmatrix} \tag{2.2}$$

$$\hat{b}(z) \equiv Tb = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \text{ and } \hat{c}(z) \equiv cT^{-1} = [b_1(z), \ldots, b_n(z)] .$$

where the polynomials a_i's and b_i's come from

$$G(s,z) = \frac{c(z)\mathrm{Adj}\,(sI-A(z))b(z)}{\det(sI-A(z))} = \frac{b_n(z)s^{n-1}+\ldots+b_1(z)}{s^n+a_n(z)s^{n-1}+\ldots+a_1(z)}$$

in which the factor coprimeness of $c\mathrm{Adj}(sI-A)b$ and $\det(sI-A)$ follows from the fact that $(A(z),b(z)) \in \mathbf{X}$.* Observe that $T(z)$ defined in (2.1) is a rational matrix in z unless $[A|b]$ is unimodular. On the other hand note

Lemma. Given $(A(z),b(z)) \in \mathbf{X}$, $T(z)$ as defined in (2.1) then $T^{-1}(z) \in \mathcal{R}^{nxn}[z]$.

Proof. By (2.2),

* Under the assumption of $\mathcal{R}(z)$-observability of the pair $(c(z), A(z))$.

$$[b, Ab, ..., A^{n-1}b]^{-1}T^{-1} = [Tb, TAb, ..., TA^{n-1}b]^{-1} = [\hat{b},\hat{A}\hat{b},...,\hat{A}^{n-1}\hat{b}]^{-1}$$

$$= \begin{bmatrix} 0 & 0 & 0 & \cdot & 1 \\ 0 & 0 & 0 & \cdot & \ell_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \ell_1 & \cdot & \cdot \\ 1 & \ell_1 & \ell_2 & \cdot & \ell_{n-1} \end{bmatrix}^{-1} \equiv L^{-1}(z)$$

where $\ell_i' s \in \mathcal{R}[z]$. Thus

$$T^{-1}(z) = [A \mid b]L^{-1}(z) \tag{2.3}$$

Notice that $L(z)$ is a unimodular matrix and so is $L^{-1}(z)$, the formula (2.3) thus completes the proof.

□

Now seek a state feedback controller

$$u(t) = k(z)x(t) + v(t) \tag{2.4}$$

with $k(z) \in \mathcal{R}^{1xn}(z)$ such that the feedback controller is physically implementable and the resulting closed-loop system has a desired coefficient vector $\alpha(z)$ in its characteristic equation. Such an implementable feedback gain $k(z)$ is called admissible. We then define $D = \{k(z) \mid k(z) \in \mathcal{R}^{1xn}(z)$ and admissible$\}$.

It is seen that $\det[sI-(A+bk)] = \det[sI-(\hat{A}+\hat{b}\hat{k})]$ where $\hat{k}(z) = k(z)T^{-1}(z)$. So if $\det[sI-(A+bk)] = s^n+\alpha(z)[1\ s\ ...\ s^{n-1}]^T$ then $\alpha(z) = a(z) - \hat{k}(z)$ where $a(z) \equiv [a_1(z)...a_n(z)]$ is the coefficient vector of the characteristic equation of the uncompensated delay system. Thus, for a specified coefficient vector $\alpha(z)$,

$$k(z) = \hat{k}(z)T(z) = (a(z)-\alpha(z))\begin{bmatrix} h_n \\ h_nA \\ \vdots \\ h_nA^{n-1} \end{bmatrix} \tag{2.5}$$

would be the desired feedback gain whenever $k(z) \in D$. In this case (2.5) and (2.6) lead to

$$k(z) = (a(z)-\alpha(z))L(z)[A \mid b]^{-1}$$

which is known as the Bass-Gura formula; also see [1] for a discussion of the extension to neutral delay systems.

Assume that

$$k(z) = w^{-1}(z)N(z) \tag{2.6}$$

is an irreducible MFD of $k(z)$ with $w(z) \in \mathcal{R}[z]$ and $N(z) \in \mathcal{R}^{1xn}[z]$. The feedback control law (2.4) then becomes

$$w(z)u(t) = N(z)x(t) + w(z)v(t) \tag{2.7}$$

To guarantee the realizability of (2.7), assume $w(0) \neq 0$. So require

$$w(0) = 1 \tag{2.8}$$

in (2.6) without loss of generality. The condition (2.8) will be referred to as the causality condition hereafter (see also [1]) and it is easy to see that the causality condition holds if and only if $k(z) \in D$. Combining (2.7) and (2.8) gives

$$u(t) = N(z)x(t) + (1-w(z))u(t) + w(z)v(t) \tag{2.9}$$

which can be implemented in a feedback-feedforward scheme and the compensated system is diagrammed in Fig. 1.

Clearly the compensated system has the mathematical model

$$\begin{aligned} w(z)\dot{x}(t) &= [w(z)A(z) + b(z)N(z)]x(t) + w(z)b(z)v(t) \\ y(t) &= c(z)x(t) \end{aligned} \tag{2.10}$$

which appears to be a neutral delay type system. However it is not difficult to see that the system (2.10) remains a retarded type of delay system because of the appearance of the feedforward part of the compensator. Actually by taking the Laplace transform of (2.10) and then letting $z = e^{-hs}$ (here $h > 0$ is the delay duration), find that the transfer function of the overall system is

$$\begin{aligned} F(s,z) &\equiv c(z)[w(z)sI - (w(z)A(z) + b(z)N(z))]^{-1}w(z)b(z) \\ &= c(z)\left[sI - (A(z) + b(z)\frac{N(z)}{w(z)})\right]^{-1}b(z) \\ &= \hat{c}(z)[sI - (\hat{A}(z) + \hat{b}(z)(a(z) - \alpha(z)))]^{-1}\hat{b}(z) \\ &= \frac{b_n(z)s^{n-1} + \ldots + b_1(z)}{s^n + \alpha_n(z)s^{n-1} + \ldots + \alpha_1(z)} \end{aligned} \tag{2.11}$$

Condition (2.8) is crucial for realizing the compensator (2.9). One may also be interested in finding a checkable criterion or a sufficient condition to guarantee that condition (2.8) holds. To do this, we write $h_n(z)$ in (2.5) as

$$h_n(z) = \left[\frac{q_1(z)}{p_1(z)} \cdots \frac{q_n(z)}{p_n(z)}\right]$$

where q_i's and p_i's are in $\mathcal{R}[z]$ and GCD $(p_i(z), q_i(z)) = 1$ for all $1 \leq i \leq n$. Denoting

$$p_0(z) = \text{least common multiple of } \{p_i(z),\ 1 \leq i \leq n\}$$

it is evident that

$$w(z) \mid p_0(z) \tag{2.12}$$

and

$$p_0(z) \mid \det[A(z) \mid b(z)] \tag{2.13}$$

since $h_n(z)$ is one of the row vectors in $[A|\ b]^{-1}$. Therefore

$$p_0(0) \neq 0\ , \tag{2.14}$$

or,

$$\det[A(0)|\ b(0)] \neq 0 \tag{2.15}$$

will imply the causality condition (2.8). In other words, the compensator (2.9) will be implementable if the pair (A(0), b(0)) is controllable. One may further notice that

$$k(z) = p_0^{-1}(z)(a(z)-\alpha(z)\begin{bmatrix} \hat{h}_n \\ \hat{h}_n A \\ \vdots \\ \hat{h}_n A^{n-1} \end{bmatrix}) \equiv p_0^{-1}(z)N_0(z) \tag{2.16}$$

where $\hat{h}_n(z) = p_0(z)h_n(z) \in \mathcal{R}^{1xn}[z]$, and that

$$p_0^{-1}(z)\begin{bmatrix} \hat{h}_n \\ \hat{h}_n A \\ \vdots \\ \hat{h}_n A^{n-1} \end{bmatrix}$$

is an irreducible MFD of $T(z)$ since $p_0^{-1}\hat{h}_n$ is an irreducible MFD of h_n, therefore condition (2.14) is also necessary for the *arbitrary* coefficient assignability of the considered system.

It should be mentioned that in the development of expression (2.11) the factor $w(z)$ was cancelled, but the dynamical behavior of the compensator (2.9) depends primarily upon the distribution of the zeros of $w(z)$. Obviously the compensator (2.9) will be stable if and only if

$$w(z) \neq 0 \qquad \text{for} \qquad |z| \leq 1 \tag{2.17}$$

Thus, by (2.12) and (2.13), the stability of the compensator will be guaranteed if

$$p_0(z) \neq 0 \qquad \text{for} \qquad |z| \leq 1 \tag{2.18}$$

or

$$\det[A(z)|\ b(z)] \neq 0 \quad \text{for} \qquad |z| \leq 1 \tag{2.19}$$

Observe that conditions (2.18) and (2.19) imply conditions (2.14) and (2.15) respectively. On the other hand, condition (2.18) is equivalent to

$$p_0(e^{-sh}) \neq 0 \qquad \text{for Re } s \geq 0 \tag{2.20}$$

plus the condition (2.14). Similarly, condition (2.19) is equivalent to

$$\det[A(e^{-sh})|\ b(e^{-sh})] \neq 0 \text{ for Re } s \geq 0 \tag{2.21}$$

along with condition (2.15)

Now summarize the above analysis as the following

Theorem. A SISO $\mathcal{R}(z)$-controllable retarded system (1.1) is arbitrarily coefficient assignable by stable feedback-feedforward compensator (2.9) if one of the following conditions holds:

(i) condition (2.18);

(ii) condition (2.20) plus condition (2.14);

(iii) condition (2.19);

(iv) condition (2.21) plus condition (2.15).

In addition, a specified coefficient vector $\alpha(z)$ can be assigned via the compensator (2.9) if and only if the corresponding w(z) determined by (2.5) and (2.6) satisfies condition (2.17).

Remark. In case the highest degree of the entries in N(z) is greater than degw(z), a division algorithm gives

$$N(z) = M(z)w(z) + R(z)$$

where M(z) and R(z) are in $\mathcal{R}^{1xn}[z]$ and the highest degree of the entries in R(z) is strictly less than deg w(z). Thus (2.7) becomes

$$\begin{aligned} u(t) &= M(z)x(t) + w^{-1}(z)R(z)x(t) + v(t) \\ &\equiv M(z)x(t) + u_1(t) \end{aligned} \tag{2.22}$$

where

$$u_1(t) = w^{-1}(z)R(z)x(t) + v(t)$$

i.e.

$$u_1(t) = R(z)x(t) + (1-w(z))u_1(t) + w(z)v(t) \tag{2.23}$$

The combination of (2.22) and (2.23) yields the compensator scheme diagrammed in Fig. 2, which is equivalent to the compensator (2.9) but will evidently save delay elements in the present situation.

III. EXAMPLES

Example 1. Consider a second order retarded system with model $\dot{x} = A(z)x + b(z)u$ having

$$A(z) = \begin{bmatrix} 2 & 2z \\ 1 & 2.5z \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.1}$$

Notice that the system is unstable [6] but $\mathcal{R}(z)$-controllable since

$$\det[A \mid b] = \det \begin{bmatrix} 1 & 2+2z \\ 1 & 1+2.5z \end{bmatrix} = .5z - 1 . \tag{3.2}$$

Next compute

$$[A|b]^{-1} = \frac{1}{.5z-1}\begin{bmatrix} 1+2.5z & -2(1+z) \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad T(z) = \frac{1}{1-0.5z}\begin{bmatrix} 1 & -1 \\ 1 & -0.5z \end{bmatrix},$$

which has an irreducible MFD

$$T(z) = \begin{bmatrix} .5z-1 & 0 \\ -0.5z & 1 \end{bmatrix}^{-1}\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}.$$

By [3], it can be claimed that the possible coefficient vectors by polynomial state feedback have the form

$$\alpha(z) = [\alpha_1, \quad \alpha_2] = a(z) + [f_1 \quad f_2]\begin{bmatrix} .5z-1 & 0 \\ -0.5z & 1 \end{bmatrix}$$

$$= a(z) + [.5z(f_1-f_2) - f_1 \quad f_2]$$

$$= [3z+.5z(f_1-f_2) - f_1, -(2+2.5z) + f_2]$$

where f_1 and f_2 are any polynomials in z. It seems not to be easy to choose f_1 and f_2 such that the resulting closed-loop system would have desired dynamical behavior or even would be stable independent of delay. However, the results presented in section II are applicable for the system (3.1) since det[A|b] defined in (3.2) does not vanish for $|z| \leq 1$. For any specified coefficient vector, e.g. $\alpha(z) = [9, 12.5]$, a simple computation gives

$$k(z) = (a-\alpha)T = \frac{1}{1-0.5z}[0.5z+23.5 \quad 1.25z^2+4.25z+9]$$

$$= [1 \quad 2.5z+13.5] + \frac{1}{1-0.5z}[24.5 \quad 22.5]$$

Thus the desired compensator is

$$u(t) = [1 \quad 2.5z+13.5]x(t) + u_1(t)$$

where

$$u_1(t) = [24.5 \quad 22.5]x(t) + 0.5z\, u_1(t) + (1-0.5z)v(t)$$

which can be implemented as suggested by Fig. 2, with only three delay elements.

Example 2. Consider another second order retarded system with model:

$$\dot{x}(t) = \begin{bmatrix} -1 & z \\ -1 & -2 \end{bmatrix} x(t) + \begin{bmatrix} z \\ 1 \end{bmatrix} u(t)$$

Note that

$$\det[sI - A(z)] = s^2 + 3s + (2+z)$$

which means [6] that the system is stable independent of delay. Suppose that one wishes to assign a specified coefficient vector $\alpha(z) = [1 \quad 2]$ via a certain type of compensator. Observe that

$$\det[A \mid b] = \det \begin{bmatrix} z & 0 \\ 1 & -(z+2) \end{bmatrix} = -z(z+2) ,$$

so that the pair (A(0), b(0)) is uncontrollable. However,

$$T(z) = \frac{1}{z(z+2)} \begin{bmatrix} 1 & -z \\ z-1 & 3z \end{bmatrix}$$

and $a(z) - \alpha(z) = [z+2 \quad 3] - [1 \quad 2] = [z+1 \quad 1]$. Thus

$$k(z) = (a - \alpha)T(z) = \frac{1}{z+2}[2 \quad -z+2] = \frac{1}{0.5z+1}[1 \quad -0.5z+1]$$

i.e.

$$w(z) = 0.5z+1$$

and

$$N(z) = [1 \quad -0.5z+1] .$$

Clearly condition (2.17) is satisfied and the desired compensator scheme is

$$u(t) = [1 \quad -0.5z+1]x(t) + 0.5zu(t) + (0.5z+1)v(t)$$

$$= x_1(t) + x_2(t) - 0.5x_2(t-h) + 0.5u(t-h) + v(t) + 0.5v(t-h)$$

in which three delay elements are needed. One may use a different scheme to save delay elements as was done before. Since

$$k(z) = \frac{1}{0.5z+1}[1 \quad -0.5z+1] = [0 \quad -1] + \frac{1}{0.5z+1}[1 \quad 2] ,$$

an alternative choice is

$$u(t) = [0 \quad -1]x(t) + u_1(t)$$

and

$$u_1(t) = [1 \quad 2]x(t) + 0.5zu_1(t) + (0.5z+1)v(t) .$$

in which only two delay elements are required.

IV. CONCLUSIONS

New types of feedback-feedforward compensation schemes were developed for coefficient assignment of SISO delay systems. These new structures reduce to the well-known state-feedback controller forms in the case when delay system under consideration is $\mathcal{R}[z]$-controllable ($w(z)=1$).

REFERENCES

[1] C. I. Byrnes, M. W. Spong, and T.-J. Tarn, "A several complex variables approach to feedback stabilization of linear neutral delay-differential systems," to appear in Math. Systems Theory.

[2] J. N. Chiasson and E. B. Lee, "Coefficient assignment by dynamic compensation for SISO retarded delay systems," submitted to 23rd CDC.

[3] E. B. Lee and W.-S. Lu, "Coefficient assignability for linear systems with delays," to appear in IEEE Trans. Automat. Contr., Dec. 1984.

[4] A. S. Morse, "Ring models for delay-differential systems," Automatica, vol. 12, pp. 529-531, 1976.

[5] E. D. Sontag, "Linear systems over commutative rings: A Survey," Ricerche di Automatica, vol. 7, pp. 1-34, 1976.

[6] E. W. Kamen, Correction to "Linear systems with commensurate time delays: stability and stabilization independent of delay," IEEE Trans. Automat. Contr., vol. AC-28, pp. 248-249, 1983.

[7] E. B. Lee and S. H. Żak, "On spectrum placement for linear time invariant delay systems," IEEE Trans. Automat. Contr., vol. AC-27, No. 2, pp. 446-449, 1982.

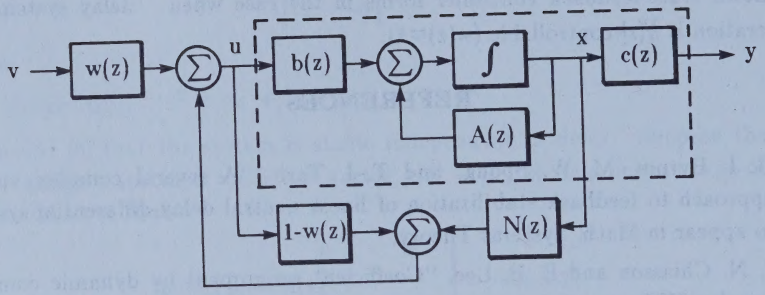


Fig. 1. Feedback-feedforward compensation scheme for linear retarded systems.

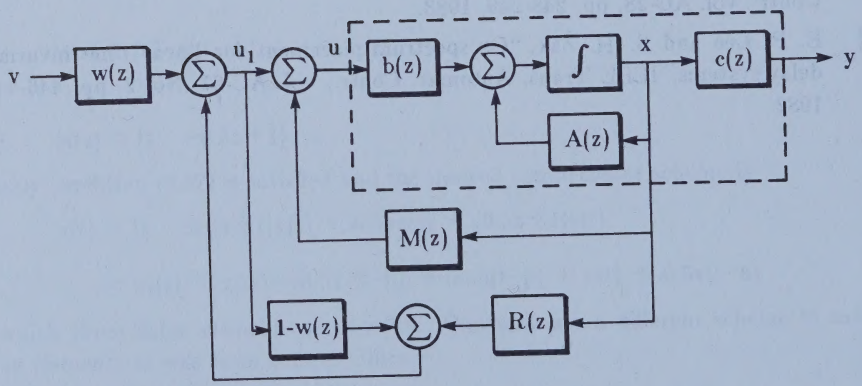


Fig. 2. Compensation scheme with the reduced number of delay lines in the controller.

SIMULTANEOUS STABILIZATION, SENSITIVITY MINIMIZATION, TRACKING AND DISTURBANCE REJECTION BY INTERPOLATION METHODS

BIJOY K. GHOSH
Department of Systems Science and Mathematics
Washington University
Saint Louis, MO 63130

Abstract: In this paper we investigate the extent to which a compensator which simultaneously renders a given r-tuple of plants internally stable can in fact be used to minimize sensitivity or track and/or reject the output of two given fixed transfer functions. The former problem of sensitivity minimization by a stabilizing compensator was addressed by Zames Francis and Helton [8], [23], [24] while the latter problem was addressed by Saeks, Murray, Chua and Karmokolias [17]. Our approach in this paper is 'interpolation-theoretic' and we obtain sufficient conditions and necessary conditions to this new class of 'simultaneous system design problems.'

1. INTRODUCTION

The "simultaneous stabilization problem" - in either discrete or continuous time - consists in answering the following question:

1.1 Given a r tuple $G_1(s)$, ..., $G_r(s)$ of pxm proper transfer functions, does there exist a compensator $K(s)$ such that the closed loop systems $G_i(s)[I + K(s)G_i(s)]^{-1}$, $i = 1, \ldots, r$ are internally stable?

As pointed out in [9], [16] the above problem arises in reliability studies and in the design of fault tolerant control systems. For a detailed reference and motivation we refer to [9]. For a basic understanding of the problem of simultaneous stabilization, we refer to [16], [20], [10]. In this paper we analyze the sensitivity minimization, disturbance rejection and tracking problems by a simultaneously stabilizing compensator.

1.2 The sensitivity minimization question that we consider in this paper may be stated as follows:

Let Ω be the set of compensators $K(s)$ which stabilizes $G_1(s)$, ..., $G_r(s)$ simultaneously. Does there exist $K(s) \in \Omega$ such that

$$S_i = \|[I + K(s)G_i(s)]^{-1} W_i\| \leq m_i \qquad (1.1)$$

for some specified positive real numbers m_1, ..., m_r and for some specified weights W_1, ..., W_r.

By a suitable definition of the norm in (1.1), the weighted sensitivity measure can be made to represent several quantities of significance (see Zames [23]). In order to describe the main result of this paper we note first of all from [4], [5], [13] that the set of r tuples of plants $G_1(s)$, ..., $G_r(s)$; deg $(G_i(s)) = n_i$, $i = 1, \ldots, r$ respectively is a manifold and is therefore a topological space. Note also that a property P is said to be 'generic' if it is true for an open and dense set of r tuples of plants. In this paper we prove the following:

Theorem 1.1: For a fixed r tuple of positive real numbers $m_1, \ldots, m_r$ and for a fixed r tuple of weights $W_1, \ldots, W_r$, a generic r tuple of pxm proper plants of degree n_i, i = 1, ..., r is simultaneously stabilizable with sensitivity $S_i \leq m_i$, i=1,...,r if

$$r \leq [\max(m, p) - 1]/\min(m, p) \tag{1.2}$$

Moreover under the above hypothesis (1.2), the stabilizing compensator may be chosen to be stable.

Note that theorem 1.1 holds for any suitable definition of the norm $\|\cdot\|$ and asserts in particular that for max(m, p) > min(m, p), a pxm proper plant can be stabilized with an arbitrary small sensitivity by a stable compensator. For a square system, which includes the case m = p = r = 1 in particular, the condition (1.2) is violated and arbitrary small sensitivity is therefore not guaranteed. In section 2 we show that under the case min(m, p) = 1, r = max(m, p) a classical Schur, Pick, Nevanlinna [14], [15] theory may be applied to provide an upper bound to the sensitivity of the r systems in the closed loop by a simultaneously stabilizing compensator. Generalization of the above procedure to the cases r > max(m, p) and/or min(m, p) > 1 is currently under investigation.

1.3 In order to describe the tracking and disturbance rejection problems we restrict our attention to a pair of single input single output plants and consider the following.

Let $n_{p_1}(s)/d_{p_1}(s)$, $n_{p_2}(s)/d_{p_2}(s)$ be a pair of single input single output plants. It is desired to construct a simultaneously stabilizing compensator $n_c(s)/d_c(s)$ which achieves either or both of the following constraints.

a) (Tracking) The output of the closed loop system $n_{p_i}(s)n_c(s)/[n_{p_i}(s)n_c(s) + d_{p_i}(s)d_c(s)]$ asymptotically tracks the impulse response of a given transfer-function $n_{t_i}(s)/d_{t_i}(s)$ for i = 1, 2, where we assume for robustness that $d_{t_i}(s)$ does not have a zero in common with $d_{p_i}(s)d_c(s)$ in the closed right half of the complex plane.

b) (Disturbance rejection) The output of the closed loop system $n_{p_i}(s)n_c(s)/[n_{p_i}(s)n_c(s) + d_{p_i}(s)d_c(s)]$ to the impulse response of a given transfer function $n_{r_i}(s)/d_{r_i}(s)$ is asymptotic to zero for i = 1, 2, where we assume for robustness that $d_{r_i}(s)$ does not have a zero in common with $n_{p_i}(s)n_c(s)$ in the closed right half of the complex plane.

We now consider the following three theorems.

Theorem 1.2: The pair of plants $n_{p_1}(s)/d_{p_1}(s)$, $n_{p_2}(s)/d_{p_2}(s)$ admit a simultaneously stabilizing compensator $n_c(s)/d_c(s)$ which satisfies the tracking condition (a) iff the total number of zeros of $n_{p_1}(s)/n_{p_2}(s)$ in between every consecutive pairs of positive real zeros (including infinity) of $d_{t_1}(s)d_{t_2}(s)(n_{p_1}d_{p_2}(s) - n_{p_2}d_{p_1}(s))$ is even.

Theorem 1.3: The pair of plants $n_{p_1}(s)/d_{p_1}(s)$, $n_{p_2}(s)/d_{p_2}(s)$ admit a simultaneously stabilizing compensator $n_c(s)/d_c(s)$ which satisfies the disturbance rejection condition (b) iff the total number of zeros of $d_{p_1}(s)/d_{p_2}(s)$ in between every consecutive pairs of positive real zeros (including infinity) of $d_{r_1}(s)\,d_{r_2}(s)(n_{p_1}\,d_{p_2} - n_{p_2}\,d_{p_1})(s)$ is even.

Theorem 1.4: Given a pair of plants $g_1(s) = n_{p_1}/d_{p_1}$ and $g_2(s) = n_{p_2}/d_{p_2}$. There exists a compensator $k(s)$ which simultaneously stabilizes $g_1(s)$, $g_2(s)$ and has the property that the closed loop systems $\psi_i(s) = g_i(s)k(s)/[1 + g_i(s)k(s)]$, $i = 1, 2$ asymptotically tracks the impulse response of a given pair of transfer functions n_{t_i}/d_{t_i}, $i = 1, 2$ respectively and asymptotically rejects the impulse response of a given pair of transfer functions n_{r_i}/d_{r_i}, $i = 1, 2$ respectively iff there exists a stable minimum phase rational function which intersects d_{p_1}/d_{p_2} at those points in the non-negative real axis including infinity where $d_{r_1}\,d_{r_2}[n_{p_1}\,d_{p_2} - n_{p_2}\,d_{p_1}]$ vanishes and intersects n_{p_1}/n_{p_2} at those points in the non-negative real axis including infinity where $d_{t_1}\,d_{t_2}(n_{p_1}\,d_{p_2} - n_{p_2}\,d_{p_1})$ vanishes. Furthermore the compensator n_c/d_c is robust iff $d_{t_i}(s)$ does not have a zero in common with $d_{p_i}(s)\,d_c(s)$ and $d_{r_i}(s)$ does not have a zero in common with $n_{p_i}(s)x\,n_c(s)$ for $i = 1, 2$ in the closed right half of the complex plane.

The above three theorems 1.2, 1.3 and 1.4 obtain the necessary and sufficient conditions for disturbance rejection and/or tracking problems for a pair of plants by a simultaneously stabilizing compensator. As depicted by Theorem 1.4, it is interesting to note the connection of the above problem with the interpolation problem originally posed and solved by Youla, Bongiorno and Lu [21].

To summarize, in this paper we propose a new design methodology which generalizes the ideas presented in [8] and [17] to the problem of 'simultaneous system design.'

2. SENSITIVITY MINIMIZATION BY A SIMULTANEOUSLY STABILIZING COMPENSATOR

Let $N_i(s)\, D_i(s)^{-1}$, $i = 1, \ldots, r$ be a r tuple of plants and let $D_c(s)^{-1}\, N_c(s)$ be the associated compensator written in coprime factorization [6] where $N_c(s)$, $D_c(s)$, $N_i(s)$, $D_i(s)$, $i = 1, \ldots, r$ are matrices of stable rational functions of appropriate order. Following Vidyasagar and Viswanadham [20], a necessary and sufficient condition for simultaneous stabilization is the solvability of the equations

$$D_c(s)\, D_i(s) + N_c(s)\, N_i(s) = \Delta_i(s) \tag{2.1}$$

for $i = 1, \ldots, r$ where $\det \Delta_i(s)$ is a stable minimum phase rational functions for all i. If $W_1, \ldots, W_r$ are weights given apriori, the sensitivity S_i defined in (1.1) is given by

$$S_i = \| D_i(s)\, \Delta_i(s)^{-1}\, D_c(s)\, W_i \| \tag{2.2}$$

2.1 Proof of Theorem 1.1: Assume $p \leq m$ without any loss of generality. We now write (2.1) in the matrix form as

$$[N_c(s)][N_1(s) \ldots N_r(s)] = [\Delta_1'(s) \ldots \Delta_r'(s)] \tag{2.3}$$

where $\Delta_i'(s) = \Delta_i(s) - D_c(s)\, D_i(s)$. For any choice of $D_c(s)$, (in particular, when $\det D_c(s)$ is stable, minimum phase) and $D_i(s)$, $\Delta_i(s)$, $i = 1, \ldots, r$; the sensitivity S_i can be made arbitrary small. Moreover if (1.2) is satisfied, (2.3) can be solved for an arbitrary right hand side for a generic r tuple of plants. To see this, let us write (2.3) as

$$\begin{bmatrix} N_{c_{j,1}} & \cdots & N_{c_{j,m-1}} \end{bmatrix} \begin{bmatrix} N^1_{1,1} & \cdots & N^1_{1,p} & \cdots & N^r_{1,1} & \cdots & N^r_{1,p} \\ & & & & & & \\ N^1_{m-1,1} & \cdots & N^1_{m-1,p} & \cdots & N^r_{m-1,1} & \cdots & N^r_{m-1,r} \end{bmatrix}$$

$$= \left[\Delta^{1'}_{j,1} - N_{c_{j,m}} N'_{m,1} \cdots \Delta^{r'}_{p,p} - N_{c_{j,m}} N^r_{m,p}\right] \tag{2.4}$$

for $j = 1, \ldots, p$ where $N_{c_{j,i}}(s)$ is the ji^{th} element of $N_c(s)$, $N^k_{i,j}(s)$ is the ij^{th} element of $N_k(s)$, $i = 1, \ldots, m$; $j = 1, \ldots, p$; $k = 1, \ldots, r$ respectively and where $\Delta^{k'}_{j,i}$ is the ji^{th} element of $\Delta_k'(s)$, $j, i = 1, \ldots, p$; $k = 1, \ldots, r$ respectively. In the worst case when $r \min(m, p) = \max(m, p) - 1$, the matrix in (2.4) is square and nonsingular for a generic r tuple of plants. This is because the set of r tuples of plants, for which the matrix in (2.4) is singular, is algebraic. By realization theory we note the existence of a plant for which the above matrix is nonsingular at $s = \infty$. Hence the above algebraic set is proper. Thus by the interpolation methods

explained in [9], [10], [11], one can choose stable rational functions $N_{c_{j,m}}(s)$, $j = 1, \ldots, p$ and solve for stable rational functions $N_{c_{j,1}}(s), \ldots, N_{c_{j,m-1}}(s)$ for $j = 1, \ldots, p$. (QED)

2.2 Let us note first of all that when r min(m, p) = max(m, p), Theorem 1.1 is no longer applicable. Hence, under this case an arbitrary small sensitivity by a simultaneously stabilizing compensator is not guaranteed. In this section, however, a classical Schur, Pick, Nevanlinna theory [14], [15] is applied to analyze the above case under the restriction min(m,p) = 1. In order to describe the main result of this section we consider the following norm of a scalar transfer function g(z)

$$||g(z)|| = \sup[|g(z)| : |z| = 1] \tag{2.5}$$

Consider the following extension of the Schur, Pick, Nevanlinna theory [14], [15], (See also Youla-Saito [22])

Proposition 2.1: Given a self-conjugate collection of pairs of complex numbers (z_j, e_j), $j = 1, \ldots, t$ with $|z_j| < 1$, there exists a rational function f(z) with real co-efficients analytic on $|z| < 1$ and continuous on $|z| \leq 1$ such that $f(z_j) = e_j$ and $||f|| < 1/k$ iff k is the largest real number such that

$$[(1 - k^2 \bar{e}_i e_j)/(1 - \bar{z}_i z_j)]_{i,j=1}^{t} \tag{2.6}$$

is non-negative definite.

Proof: To say that f(z) interpolates the pair of complex numbers (z_j, e_j) is to say that kf(z) interpolates the pair $(z_j, k e_j)$ for some $k \in \mathbb{R}$. The proof now follows from the interpolation results of Youla-Saito [22]. (QED)

Let us now consider a m tuples of 1xm proper plants given by

$$[d_i(s)]^{-1} [n_{i1}(s) \ldots n_{im}(s)] \tag{2.7}$$

for i = 1, ..., m together with a mx1 proper compensator

$$[n_{c_1}(s) \ldots n_{c_m}(s)]^T [d_c(s)]^{-1} \tag{2.8}$$

The associated closed loop system is stable (see [19], [20]) provided there exists stable, minimum phase rational functions $\Delta_1(s), \ldots, \Delta_r(s)$ such that

$$N(s)\underline{n}_c = [\Delta_1 - d_c\, d_1 \ldots \Delta_m - d_c\, d_m]^T \tag{2.9}$$

where $n_{ij}(s)$ is the ij^{th} entry of the mxm matrix N(s) and where

$$\underline{n}_c(s) = [n_{c_1}(s) \ldots n_{c_m}(s)]^T. \tag{2.10}$$

The problem of simultaneous stabilization (see [9], [10]) is to obtain $d_c(s)$, $\Delta_i(s)$ $i = 1, \ldots, m$ such that $\underline{n}_c(s)$ as given by (2.10) is a vector

of stable rational functions. Let $s_1, \ldots, s_t$ be the set of points in the closed right half of the complex plane where det N(s) vanishes. For a generic m tuple of plants, s_i, i = 1, ..., t may be assumed to be simple so that adj $N(s_i)$ are rank 1 matrices. Define

$$\delta_i = [\delta_{i1} \cdots \delta_{im}]^T \tag{2.11}$$

to be a vector in the span of adj $N(s_i)$. In order that (2.9) may be solved for a stable n_c one needs to interpolate the points (s_i, e_i) i = 1, ..., t by a stable rational function $d_c(s)$ where

$$e_i = \left. \frac{\Delta_1 \delta_{i1} + \ldots + \Delta_m \delta_{im}}{d_1 \delta_{i1} + \ldots + d_m \delta_{im}} \right|_{s=s_i} \tag{2.12}$$

Let us transform the complex s domain to the z domain by the conformal transformation

$$z = (1 + s)/(1 - s) \tag{2.13}$$

which maps the closed right half plane to the closed unit disc, we obtain a mapping with transforms the pair

$$(s_i, e_i) \to \left(\frac{s_i - 1}{s_i + 1}, e_i \right) \triangleq (z_i, e_i) \tag{2.14}$$

for i = 1, ..., t. We now have the following theorem

> Theorem 2.2: Given a generic m tuple of 1xm plants (2.7), there exists a simultaneously stabilizing compensator (2.8) such that the associated sensitivity is upper bounded by
>
> $$S_i \leq ||W||\ ||d_i||\ k^{-1} \tag{2.15}$$
>
> for i = 1, ..., m; where k is the largest real number such that (2.6) is non-negative definite.

Proof: Assume $\Delta_i(s) \equiv 1$ i = 1, ..., m. By proposition 2.1, there exists $d_c(z)$ such that $||d_c(z)|| \leq 1/k$ and which interpolates the points (z_i, e_i) i = 1, ..., t given by (2.14). Clearly $d_c\left(\frac{1+z}{1-z}\right)$ would interpolate the points (s_i, e_i) i = 1, ..., t. Moreover note that the associated sensitivity

$$S_i = ||W\, d_c(s)\, d_i(s)/\Delta_i(s)|| \leq ||W||\ ||d_i||\ k^{-1} \qquad \text{(QED)}$$

Remark: For a constructive algorithm to obtain a stable rational function of minimum norm which satisfies a given interpolation condition we refer to [8], [24].

2.3 In section 2.2, the choice of $\Delta_i(s) \equiv 1$ was made rather arbitrary. In this section we assume r = 2, min(m, p) = 1, max(m, p) = 2 and show that one can choose $\Delta_1(s)$, $\Delta_2(s)$ in order that the sensitivity functions

$$S_i(s) = d_i(s)\, d_c(s)/\Delta_i(s) \tag{2.16}$$

are equal in magnitude at some specified complex conjugate points $s = s_j'$ $j = 1, \ldots, t_1$. Roughly speaking, this amounts to saying that the sensitivity functions are matched at a finite set of points. Furthermore in order to reduce the sensitivities, we have the following interpolation problem.

> Assume $\Delta_1(s) \equiv 1$ without any loss of generality. Construct a stable minimum phase rational function $\Delta_2(s)^{-1} = \Delta^*(s)^{-1}$ of minimum norm which interpolates the points $(s_i', |d_2/d_1(s_i')|^{-1})$, for all $i = 1, \ldots, t_1$

The above interpolation problem can be solved by the application of the following proposition.

> Proposition 2.3: (Ball and Helton [3]) Given collections of complex numbers $\{z_j\}$ and $\{w_j\}$ with $|z_j| < 1$ and $w_j \neq 0$ $(j = 1), \ldots, n)$, there exists a function f, analytic on $\{|z| < 1\}$, such that $f(z_k)=w_k$, $k = 1, \ldots, n$ and $f(z) \neq 0$ for $|z| < 1$ with $||f|| \leq 1$ iff for some choice of integers $\{n_k\}_1^m$ the matrix
>
> $$\Lambda_{\{n_k\}} \triangleq \left[\frac{-\ell n w_k - \overline{\ell n w_\ell} + 2\pi i(n_\ell - n_k)}{1 - z_k \bar{z}_\ell} \right]_{k,\ell=1}^{n} \tag{2.17}$$
>
> is positive definite.

Thus by the technique used in proposition 2.1 one can scale f by a constant k and thereby obtain f of minimum norm satisfying the interpolation constraint. In order to construct f we refer to [7], [1], [2]. Note that f obtained this way is no longer rational but can be chosen to be a symmetric function, i.e. $\overline{f(s)} = f(\bar{s})$. By substituting $\Delta_1(s) \equiv 1$ and $\Delta_2(s) = 1/f(s)$ in (2.13) for $m = 2$, one can now apply the procedure described in section 2.2 in order to construct $d_c(s)$. Thus we have an analytic function f(s) and a stable rational function $d_c(s)$ which we now substitute in (2.10) to compute $n_{c_1}(s)$ and $n_{c_2}(s)$. The functions $n_{c_1}(s)$, $n_{c_2}(s)$ are no longer rational but can be approximated arbitrary closely by stable rational functions (see [18] for details).

3. Proof of Theorem 1.2: A necessary and sufficient condition that the pair of plants is simultaneously stabilizable is the existence of a stable, minimum phase rational functions $\Delta_1(s)$, $\Delta_2(s)$ such that

$$n_{p_i}\, n_c + d_{p_i}\, d_c = \Delta_i, \quad i = 1, 2. \tag{3.1}$$

On the other hand a necessary and sufficient condition that the tracking condition (a) is satisfied (see [17] for details) is the existence of stable rational functions $\Delta_1(s)$, $\Delta_2(s)$ such that

$$n_{t_i} d_{p_i} d_c = \delta_i d_{t_i} \Delta_i, \quad i = 1, 2 \tag{3.2}$$

This is so because for tracking we want

$$n_{t_i}/d_{t_i}[1 - n_{p_i} n_c/(n_{p_i} n_c + d_{p_i} d_c)] \tag{3.3}$$

to be a stable rational function $\delta_i(s)$. Solving (3.1) for $d_c(s)$ we have

$$d_c(s) = [\Delta_2 n_{p_1} - \Delta_1 n_{p_2}]/[n_{p_1} d_{p_2} - n_{p_2} d_{p_1}] \tag{3.4}$$

Substituting $d_c(s)$ from (3.4) into (3.2) we have

$$\frac{n_{t_i} d_{p_i}[\Delta_2 n_{p_1} - \Delta_1 n_{p_2}]}{(n_{p_1} d_{p_2} - n_{p_2} d_{p_1}) d_{t_i}} = \delta_i \Delta_i \tag{3.5}$$

$i = 1, 2$. In order that δ_1, δ_2 be stable we have the following:

"Δ_2/Δ_1 intersects n_{p_2}/n_{p_1} at those points in the closed right half plane including infinity where $(n_{p_1} d_{p_2} - n_{p_2} d_{p_1})$ or d_{t_1} or d_{t_2} vanish. Finally the proof follows by an application of the interpolation lemma due to Youla, Bongiorno and Lu [21].

(QED)

Proof of Theorem 1.3: A necessary and sufficient condition that the disturbance rejection condition is satisfied (see [17] for details) is the existence of stable rational functions $\delta_3(s)$, $\delta_4(s)$ such that

$$n_{r_i}/d_{r_i}[n_{p_i} n_c/(n_{p_i} n_c + d_{p_i} d_c)] = \delta_{i+2} \tag{3.6}$$

$i = 1, 2$. Equivalently

$$n_{r_i} n_{p_i} n_c = \delta_{i+2}(s) d_{r_i}(s) \Delta_i(s), \quad i = 1, 2. \tag{3.7}$$

Solving for $n_c(s)$ from (3.1) and substituting back in (3.7) the proof is now similar to that of theorem 1.2.

(QED)

Proof of Theorem 1.4: In this case one needs to solve (3.1), (3.2) and (3.7) simultaneously. Solving for both n_c and d_c from (3.1) and substituting into (3.2) and (3.7), the theorem follows by arguments similar to that of theorem 1.2. The robustness result follows by analysing (3.3) and (3.6). In fact if d_{r_i} has a root in common with $n_{p_1} d_c$ in the right half plane, such a root would appear as a pole of $\delta_{i+2}(s)$ for arbitrary small perturbation of the plant thus making $\delta_{i+2}(s)$ unstable.

(QED)

REFERENCES

[1] Adamjan, V. M., Arov, D. Z. and Krein, M. G., "Analytic properties of Schmidt pairs for a Hankel operator and the generalized Schur-Takagi problem," Math. USSR Sbornik, Vol. 15, 1971, No. 1, pp. 31-73 (English).

[2] _____, "Infinite block Hankel matrices and their connection with the interpolation problem," AMS Translations Ser 2, Vol. 111 (1978).

[3] Ball, J. A. and Helton, J. W., "Interpolation with outer functions and gain equalization in amplifiers," Prof. of the Internat. Conf. on Math. Theory of Circuits and Systems (Delft, July 1979).

[4] Byrnes, C. I. and Hurt, N. E., "On the moduli of linear dynamical systems," Adv. in Math., Suppl. Series, Vol. 4, pp. 83-122, 1978.

[5] Clark, J. M. C., "The consistent selection of local co-ordinates in linear system identification," Proc. JACC, Purdue, pp. 576-580, 1976.

[6] Desoer, C. A., Liu, R. W., Murray, J. W. and Saeks, R., "Feedback system design: the fractional representation approach to analysis and synthesis," IEEE Trans. Automat. Contr., Vol. AC-25, pp. 401-412, 1980.

[7] Donoghue, W. F., Monotone matrix functions and analytic continuation, New York, Springer, 1974.

[8] Francis, B. A., Helton, J. W. and Zames, G., "Optimal minimax feedback controllers for linear multivariable systems," September, 1982, (preprint).

[9] Ghosh, B. K., Simultaneous stabilization and pole placement of a multimode linear dynamical system, Ph.D. Dissertation, Harvard University, USA, 1983.

[10] _____, "Simultaneous partial pole placement-a new approach to multimode system design." (Submitted to IEEE Trans. on Automat. Contr.)

[11] _____, "Transcendental and interpolation methods in simultaneous stabilization and simultaneous partial pole placement problems." (Submitted to SIAM J. of Control and Optimization.)

[12] Ghosh, B. K. and Byrnes, C. I., "Simultaneous stabilization and simultaneous pole-placement by non-switching dynamic compensation," IEEE Trans. Automat. Contr., Vol. AC-28, pp. 735-741, 1983.

[13] Hazewinkel, M. and Kalman, R. E., "On invariants, canonical forms and moduli for linear constant finite-dimensional dynamical system," in Lecture Notes in Econo.-Math. System Theory, Springer Verlag, Berlin, Vol. 131, pp. 48-60, 1976.

[14] Nevanlinna, R., "Über beschrankte Funktionen, die in gegebenen Punkten vorgeschriebene werte annehmen," Ann. Acad. Sci. Fenn. 13, No. 1, 1919.

[15] Pick, G., Über die Beschränkunger analytischer Funktionen, Welche durch vorgegebenen Funktionswerte bewiskt sind, Math. Ann. 77, 7-23, 1916.

[16] Saeks, R. and Murray, J. J., "Fractional representation, algebraic geometry, and the simultaneous stabilization problem," IEEE Trans. Automat. Contr., Vol. AC-27, pp. 895-903, 1982.

[17] Saeks, R., Murray, J. J., Chua, O. and Karmokolias, C., "Feedback system design: the single variate case," unpublished notes, Texas Tech. Univ., Lubbock, Texas, 1980.

[18] Vidyasagar, M. and Davidson, K. R., "A characterization of all stable stabilizing compensators for single input single output systems," (to be published).

[19] Vidyasagar, M. Schneider, H. and Francis, B., "Algebraic and topological aspects of feedback stabilization," IEEE Trans. Automat. Contr., Vol. AC-27, pp. 880-894, 1982.

[20] Vidyasagar, M. and Viswanadham, N., "Algebraic design techniques for reliable stabilization," IEEE Trans. Automat. Contr., Vol. AC-27, pp-1085-1095, 1982.

[21] Youla, D. Bongiorno, J. and Lu, C., "Single loop feedback stabilization of linear multivariable dynamic plant," Automatica, 10, pp. 155-173, 1974.

[22] Youla, D. Saito, M., "Interpolation with positive-real functions," Journal of the Franklin Institute 284, 77-108, 1967.

[23] Zames, G., "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms and approximate inverses," IEEE Trans. Automat. Contr., Vol. AC-20, 1981.

[24] Zames, G. and Francis, B. A., "Feedback and minimax sensitivity," Adv. Group for Aerospace research and development, NATO Lecture Notes, 117, Multivariable analysis and design techniques.

COMPENSATOR PARAMETRIZATION DESIGN FOR MULTIVARIABLE SYSTEMS

SHOU-YUAN ZHANG

Department of Electrical Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794

ABSTRACT

In this paper, a new parametrization of compensators to achieve an arbitrary denominator of the multivariable feedback systems is discussed. The parametrization insures the properness of the compensators and the parametrization coefficients appear to be the coefficients of the compensators. By using the partial dynamic decoupling, which can always be achieved by the parametrization approach, some design requirements such as steady-state error constants and sensitivity optimization are satisfied. These design requirements can not be achieved by the usual model control techniques.

I. INTRODUCTION

It is well known that one of the most important approaches in linear systems design is the model control, with this approach a compensator will be found to achieve a desired denominator of the feedback system. While the denominator can be arbitrarily assigned under some reasonable conditions, the numerator of the feedback system is left open in control. Hence, some design requirements such as steady-state error constants, sensitivity optimization and decoupling, etc., can not be achieved by using the model control approach. It was, however, found that a family of compensators can be used to achieve a same denominator of the feedback system. Consider a proper plant with coprime fractions

$$G(s) = N(s)D^{-1}(s) = \bar{D}^{-1}(s)\bar{N}(s)$$

Let X(s) and Y(s) be polynomial matrices such that

$$X(s)D(s) + Y(s)N(s) = I$$

Let M(s) be an arbitrary polynomial matrix, then

$$C(s) = [F(s)X(s)-M(s)\bar{N}(s)]^{-1}[F(s)Y(s)+M(s)\bar{D}(s)] \qquad (1)$$

is the representation of the family of compensators for the denominator F(s) of the feedback system. This fact was first found in [1] and was further explored in [2-4]. It is noted that a proper choice of compensators in this family could be helpful in obtaining the specified error constants and the sensitivity optimization, e.g., in [5] for single-input-output systems. The disadvantage of the parametrization of compensators (1) is that the properness of the compensators is not insured. In [6], a new parametrization of compensators was presented and used for single-input-output systems. The properness of compensators is built in the parametrization and the parametrization coefficients appear to be the coefficients of compensators. In this paper, the parametrization will be extended into the multivariable systems design. First, we use the parametrization to achieve a partial dynamic decoupling, then to achieve the expected steady-state error constants and sensitivity optimization. The design is completed by choosing the free coefficients of the compensators. An example will be given to illustrate the design.

II. PARAMETRIZATION

Consider the unity feedback system shown in Fig. 1, where the qxp rational matrix

$$G(s) = N(s)D^{-1}(s)$$

is the transfer function of the proper plant and

$$C(s) = H^{-1}(s)K(s)$$

is the pxq compensator. The transfer function of the feedback system can be written as

$$G_f(s) = N(s)F^{-1}(s)K(s)$$

where

$$F(s) = H(s)D(s)+K(s)N(s) \tag{2}$$

Let $\delta_{ci}D(s)$ denote the ith column degree of D(s). It is assumed that

$$\delta_{ci}D(s) = \mu_i, \quad \mu = \text{Max}\,\mu_i, \qquad i = 1,2,\ldots,p.$$

We write

$$D(s) = D_\mu s^\mu + \ldots + D_1 s + D_0$$

$$N(s) = N_\mu s^\mu + \ldots + N_1 s + N_0$$

and define the generalized resultant of {D(s),N(s)} as follows:

$$S_m = \begin{bmatrix} D_0 & D_1 & \cdots & \cdots & D_\mu & & \\ N_0 & N_1 & \cdots & \cdots & N_\mu & & \\ & D_0 & \cdots & \cdots & \cdots & D_\mu & \\ & N_0 & \cdots & \cdots & \cdots & N_\mu & \\ & & & \ddots & & & \\ & & D_0 & \cdots & \cdots & \cdots & D_\mu \\ & & N_0 & \cdots & \cdots & \cdots & N_\mu \end{bmatrix} \begin{matrix} \} \text{ 0th block} \\ \\ \\ \\ \\ \} \text{ mth block} \end{matrix}$$

Then the equation (2) can be written as the following equation:

$$[H_0\ K_0\ H_1\ K_1\ \ldots\ H_m\ K_m]S_m = \bar{C}S_m = [F_0\ F_1\ \ldots\ F_{\mu+m}] = \bar{F} \tag{3}$$

where

$$F(s) = F_{\mu+m}s^{\mu+m} + \ldots + F_1 s + F_0$$

is the denominator matrix of the feedback system.

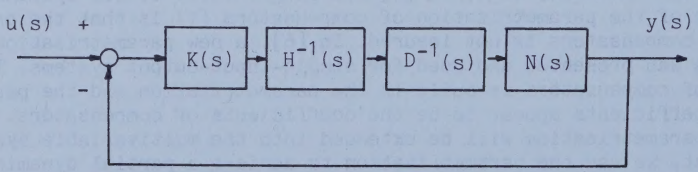


Fig. 1

The following result can be used for the purpose of model control.

Proposition 1 [7]:

Let the strictly proper plant $G(s)=N(s)D^{-1}(s)$ be given, where $D(s)$ and $N(s)$ are right coprime and $D(s)$ is column reduced with $\delta_{ci}D(s)=\mu_i$. Let ν be the observerbility index of $G(s)$. If $F(s)$ is row-column reduced [4]with ith column degree μ_i and jth row degree m_j, $m_j \geq \nu-1$ for all j, then there exists a proper compensator $C(s)=H^{-1}(s)K(s)$ to meet (2), where $H(s)$ is row reduced with jth row degree m_j, i=1,2,...,p and j=1,2,...,q.

It is shown in [8] that after deleting the zero columns the matrix S_{m_j} will have a full column rank if $m_j \geq \nu-1$. Thus, Proposition 1 can be proved by using this result. We note that equation (2) can be solved row by row, i.e., for every row of $\bar{F}$, we find the corresponding row of $\bar{C}$. Hence, we consider the following equation:

$$\bar{C}^j S_{m_j} = \bar{F}^j \tag{4}$$

where $\bar{C}^j$ and $\bar{F}^j$ are the jth row of $\bar{C}$ and $\bar{F}$ respectively. Let $\bar{F}^j$ be given, the solution $\bar{C}^j$ can be partitioned into two parts. One is called the particular solution, which constitutes a proper compensator to achieve $F(s)$ and has been intensively discussed, e.g., in [3,4,7,9]. Since there are $(m_j+1)q-n$ dependent rows in S_{m_j}, there exists a left null space of S_{m_j} with dimension $(m_j+1)q-n$, where n is the degree of $G(s)$. By using this null space, we can find a solution for the homogeneous part of equation (2), which is the another part of the solution of (4). Thus, the general solution of (4) can be written as

$$\bar{C}^j = \bar{C}_p^j + [r_1\ r_2 \cdots r_{\alpha_j}][-B_0\ \ A_0 \cdots -B_{m_j}\ \ A_{m_j}] \tag{5}$$

where $\bar{C}_p^j$ is the particular solution of (4), $\alpha_j=(m_j+1)q-n$ and $[-B_0\ \ A_0 \cdots -B_{m_j}\ \ A_{m_j}]$ is a basis of the left null space of S_{m_j}. For a given $\bar{F}^j$, $\bar{C}^j$ in (5) is a complete parametrization of the jth row of $C(s)$, where $C(s)$ is the proper compensator to meet (2). Thus, a complete parametrization of $C(s)$ to meet (2) has been established. Since this parametrization is built under the condition of $H(s)$ row reduced with jth row degree m_j and the corresponding row degree of $K(s)$ is less than or equal to m_j, j=1,2,...,q, the properness of the compensators is insured in the parametrization. The number of indetermined parameters, $(m_j+1)q-n$, depends on the degree of the compensators. Hence we have the following result.

Proposition 2:

Consider the design problem in Proposition 1, if $m_j \geq \nu-1$ for all j, in addition to achieve an arbitrary denominator $F(s)$, there exist $\sum (m_j+1)q-nq$ arbitrary assignable parameters in choosing the coefficients of compensator to meet (2).

Since the denominator of the feedback system has been given, Proposition 2 means that the numerator of the feedback system can be partially assigned. This fact will be useful in achieving some design requirements other than the pole assignment.

III. MULTIVARIABLE UNITY FEEDBACK SYSTEM DESIGN

In this section, we discuss the design of multivariable unity feedback system design by using the compensator parametrization presented in Section II.

Consider the pxp plant $G(s)=N(s)D^{-1}(s)$. We assume first that the

matrix N(s) is nonsingular and Hurwitz. If we choose $F(s)=\hat{F}(s)N(s)$, the transfer function of the feedback system is

$$G_f(s)=N(s)[\hat{F}(s)N(s)]^{-1}K(s)=\hat{F}^{-1}(s)K(s)$$

where $\hat{F}(s)$ is an arbitrary diagonal polynomial matrix such that there exists a unimodular matrix U(s) and $U(s)\hat{F}(s)N(s)$ is row-column reduced with column degrees α_i, i=1,2,...,p and row degrees m_j, $m_j \geq \nu-1$ for all j. Note that we have the following equation from (2):

$$H(s)D(s)+K(s)N(s)=U(s)\hat{F}(s)N(s)$$

where $C(s)=H^{-1}(s)K(s)$ is a proper compensator. This compensator can also, however, be obtained from the following equation:

$$U^{-1}(s)H(s)D(s)+U^{-1}(s)K(s)N(s)=\hat{F}(s)N(s)$$

with $C(s)=[U^{-1}(s)H(s)]^{-1}U^{-1}(s)K(s)=H^{-1}(s)K(s)$.

For simplicity, we assume that the plant is 2x2 and hence the transfer function of the feedback system is

$$G_f(s)=\hat{F}^{-1}(s)K(s)=\begin{bmatrix}\hat{f}_1(s) & 0\\ 0 & \hat{f}_2(s)\end{bmatrix}^{-1}\begin{bmatrix}k_{11}(s) & k_{12}(s)\\ k_{21}(s) & k_{22}(s)\end{bmatrix}$$

where

$$\hat{f}_i(s)=\hat{f}_{i\alpha_i}s^{\alpha_i}+\ldots+\hat{f}_{i0}, \qquad i=1,2$$

$$k_{ij}(s)=k_{ij\beta_{ij}}s^{\beta_{ij}}+\ldots+k_{ij0}, \qquad i,j=1,2$$

with

$$\alpha_i \geq \beta_{ij} \qquad i,j=1,2$$

Hence, the ith output of the feedback system is

$$y_i(s)=k_{i1}(s)\hat{f}_i^{-1}(s)u_1(s)+k_{i2}(s)\hat{f}_i^{-1}(s)u_2(s), \qquad i=1,2 \qquad (6)$$

where

$$u(s)=[u_1(s) \quad u_2(s)]'$$

is the input. Let $k_{ij0}=0$, $i\neq j$, the static decoupling [10] is achieved. The partial dynamic decoupling, which is different from the dynamic decoupling, can be achieved if we let $k_{ij\theta}=0$, i=j, $\theta=1,2,\ldots$. We show that if the degree of the compensator is large enough the partial dynamic decoupling can always be achieved. Note that the denominator of the feedback system in our case is $\hat{F}(s)N(s)$ with $\hat{F}(s)$ diagonal. Since N(s) is Hurwitz, then N_0 should be nonsingular in order to insure that no zeros of detN(s) are located at the origin of the s-plane. Hence, the first D block in the resultant S_m can always be dependent on other blocks if m is large enough. Consider the equation

$$H_0D_0+K_0N_0=F_0N_0$$

Since F_0 is a nonzero diagonal matrix, if we assign H_0 to be zero, it appears that K_0 will be a nonzero diagonal matrix. Thus, the static decoupling can always be achieved. Consider the shifting property of S_m, the partial dynamic decoupling is proved to be achievable. Once the static or partial dynamic decoupling is achieved, the steady-state error constants may be assigned. It is proved in [11] that a type $[m_1,m_2]$ plant makes the stable feedback system automatically follows the signal $[s^{-\eta_1},s^{-\eta_2}]'$ without steady state error, where $1 \leq \eta_i \leq m_i$, i=1,2. For instance, if the plant is type [2,2], then the

following equations are satisfied automatically:

$$k_{120}=k_{121}=k_{210}=k_{211}=0$$

$$k_{110}=\hat{f}_{10},\quad k_{111}=\hat{f}_{11},\quad k_{220}=\hat{f}_{20},\quad k_{221}=\hat{f}_{21}$$

It is clear that in this case a partial dynamic decoupling has been automatically achieved. Thus, to obtain a specified acceleration error constant E_{2i}, we solve the following equation:

$$E_{2i}=\lim_{s\to 0}\frac{s^2\hat{f}_i(s)}{\hat{f}_i(s)-k_{ii}(s)}=\frac{\hat{f}_{i0}}{\hat{f}_{i2}-k_{ii2}} \tag{7}$$

and have

$$k_{ii2}=\hat{f}_{i2}-\hat{f}_{i0}/E_{2i}$$

Note that to endow the acceleration error constant E_{2i} with a practical meaning, if $u_1(s)$ and $u_2(s)$ are comparable, for instance, we need to let

$$k_{123}=k_{213}=0$$

i.e., to achieve a deeper partial dynamic decoupling.

Following [12], we use the minimization of the integral

$$I=\frac{1}{2\pi i}\int_{-j\infty}^{j\infty}\frac{S(s)S(-s)}{-s^2}\,ds \tag{8}$$

to achieve the minimization of the sensitivity, where S(s) is the sensitivity function,

$$S(s)=[I+G(s)C(s)]^{-1}=I-G_f(s)=\begin{bmatrix}1-k_{11}(s)f_1^{-1}(s) & -k_{12}(s)f_1^{-1}(s)\\ -k_{21}(s)f_2^{-1}(s) & 1-k_{22}(s)f_2^{-1}(s)\end{bmatrix} \tag{9}$$

In the practical design, the feedback system $G_f(s)$ is always strictly proper. Hence, the norms of the diagonal elements of S(s) are much larger than the others in the high frequency range. Therefore, if we minimize the nondiagonal elements in the low frequency range, say, by partial dynamic decoupling, the sensitivity function is diagonally dominant. Thus, the technique presented for single-input single-output systems [12] can be employed.

For the case of N(s) not Hurwitz, the unstable modes of N(s) can separated and removed from N(s), see [13], then, proceed the design.

IV. EXAMPLE

Consider a strictly proper plant

$$G(s)=N(s)D^{-1}(s)=\begin{bmatrix}-s & 1\\ 1 & s+1\end{bmatrix}\begin{bmatrix}s^2 & 0\\ 0 & s^2\end{bmatrix}^{-1}$$

where D(s) and N(s) are right coprime, D(s) is column reduced with column degrees 2, the system G(s) is type [2,2]. The design requirements are as follows:

i. Stabilize the system and place the poles of the feedback system at -1 in s-plane if possible.

ii. The acceleration error constants E_{2i}, i=1,2, should be larger than or equal to 1/4.

iii. Minimize the sensitivity function (9).

First, we form the resultant of D(s) and N(s) and find the observerbility index of G(s), $\nu=2$, see [8]. Since we have two more design requirements other than the pole assignment, we need to find an $\hat{F}(s)$ such that there exists a unimodular matrix U(s), $U(s)\hat{F}(s)N(s)$ is row-column reduced with column degrees $\mu_1=\mu_2=2$ and row degrees $m_1=m_2=\nu-1+2=3$. Hence, we choose

$$\hat{F}(s)=\begin{bmatrix}(s+1)^4 & 0\\ 0 & (s+1)^4\end{bmatrix}$$

Then, we have

$$\hat{F}(s)N(s)=\begin{bmatrix}-s(s+1)^4 & (s+1)^4\\ (s+1)^4 & (s+1)^5\end{bmatrix}$$

We form the resultant S_3, then solve the equation (3) to find a particular solution, we also find a basis of the left null space of S_3 by searching [8]. Thus the general solution of (3) is

$$\bar{C}=\bar{C}_p+\begin{bmatrix}r_1 & r_2 & r_3 & r_4\\ r_1' & r_2' & r_3' & r_4'\end{bmatrix}[-B_0\ A_0\ -B_1\ A_1\ -B_2\ A_2\ -B_3\ A_3]=$$

$$=\begin{bmatrix}0 & 1 & -1 & 4 & -4 & 6 & -6 & 4 & -4 & 1 & -1 & 0\\ 1 & 1 & 4 & 5 & 6 & 10 & 4 & 10 & 1 & 5 & 0 & \mathbf{1}\end{bmatrix}+$$

$$+\begin{bmatrix}r_1 & r_2 & r_3 & r_4\\ r_1' & r_2' & r_3' & r_4'\end{bmatrix}\begin{bmatrix}0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ -1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}$$

It is of interest to find the general solution to achieve the closed loop denominator $\hat{F}(s)N(s)$ as follows:

$$C(s)=H^{-1}(s)K(s)$$

where H(s) and K(s) are expressed by the complete parametrization form:

$$H(s)=\begin{bmatrix}-s^3-(4-r_3)s^2-(6-r_1+r_4)s & (1-r_4)s^2+(4-r_2-r_3-r_4)s+6-r_1-r_2\\ (1+r_3')s^2+(4+r_1'-r_4')s+(6-r_2') & s^3+(5-r_4')s^2+(10-r_2'-r_3'-r_4')s+6-r_1'-r_2'\end{bmatrix}$$

$$K(s)=\begin{bmatrix}r_3s^3+r_1s^2+4s+1 & r_4s^3+r_2s^2\\ r_3's^3+r_1's^2 & r_4's^3+r_2's^2+4s+1\end{bmatrix}$$

Since H(s) will not appear explicitly on the transfer function of the feedback system, we need only to consider the matrix K(s). We consider first the first row of K(s). To satisfy the requirement ii, we have the following from equation (7):

$$k_{112}=f_{12}-f_{10}/E_{21}=4-1/\tfrac{1}{4}=0$$

i.e.,

$$r_1=0$$

At same time, we let $r_2=0$. Now, the first row of K(s) is

$K^1(s)=[r_3s^3+4s+1 \quad r_4s^3]$

To satisfy the requirement iii, we let $r_4=0$ in order to insure (9) diagonally dominant. Then, we find the diagonal element of S(s) in (9) to be

$$1-k_{11}(s)f_1^{-1}(s)=\frac{s^4+(4-r_3)s^3+6s^2}{s^4+4s^3+6s^2+4s+1}$$

To minimize (8), after some computations, we have

$$r_3=4$$

Following the same procedure, we find the solution for second row of K(s). Thus the solution of this example is as follows:

$$C(s)=H^{-1}(s)K(s)=\begin{bmatrix}-s^3-2s & -3s^2+8s+6\\ s^2+6 & s^3+s^2+6s+6\end{bmatrix}^{-1}\begin{bmatrix}4s^3+4s+1 & 0\\ 0 & 4s^3+4s+1\end{bmatrix}.$$

REFERENCES:

1. D.C.Youla, J.J.Bongiorno and H.A.Jabr, "Modern Wiener-Hopf Design of Optimal Controllers, Part I, Part II," IEEE Trans. Automat. Contr., Vol. AC-21, 1976.
2. V.Kucera, "Discrete Linear Controls," John Wiley & Sons, New York, 1979.
3. H.H.Rosenbrock and G.E.Hayton, "The General Problem of Pole Placement," Int. J. Contr., Vol. 27, 1978.
4. F.M.Callier and C.A.Desoer, "Multivariable Feedback Systems," Springer-Verlag, New York, 1982.
5. J.J.Bongiorno and D.C.Youla, "On the Design of Single-loop Single-input-output Feedback Control Systems in the Complex-Frequency Domain," IEEE Trans. Automat. Contr., Vol. AC-22, 1977.
6. S.Y.Zhang and C.T.Chen, "A Note On Parametrization," Submitted.
7. S.Y.Zhang, "On Solving the Polynomial Diophantine Equation," Proc. 23rd IEEE Conf. Decision and Control, 1984.
8. C.T.Chen, "Linear System Theory and Design," Holt, Reinhart and Winston, New York, 1984.
9. E.Emre, "The Polynomial Equation $\Theta\Theta_C+RP_C=\phi$ with Applications to Dynamic Feedback Systems," SIAM J. Contr. Optimiz., Vol. 18, 1980.
10. W.A.Wolovich, "Linear Multivariable Systems," Springer Verlag, New York, 1974.
11. C.A.Wolfe and J.S.Meditch, "Theory of System Type for Linear Multibariable Servomechanism," IEEE Trans. Automat. Contr., Vol. AC-22, 1977.
12. G.C.Newton et. al., "Analytical Design of Linear Feedback Controls," New York, John Wiley & Sons, 1957.
13. C.A.Desoer and M.J.Chen, "Design of Multivariable Feedback Systems with Stable Plant," IEEE Trans. Automat. Contr., Vol. AC-26, 1981.

COMPUTING THE ESTIMATOR-CONTROLLER FORM OF A COMPENSATOR

DOUGLAS J. BENDER and RICHARD A. FOWELL

Hughes Aircraft Company, Space and Comm. Group, El Segundo, CA

ABSTRACT

A technique is presented for finding the estimator and controller gains to implement a desired compensator in estimator-controller form. The technique is derived by mapping the states of the compensator to those of the estimator-controller using a linear transformation. Using the technique requires finding the real invertible solutions of a nonsymmetric steady state matrix Riccati equation. The Riccati equation may have zero, two or more real invertible solutions. From each solution a set of estimator and controller gains can be computed. Multiple solutions correspond to freedom to choose which closed loop system poles are associated with the controller and which are associated with the estimator. This technique can be applied to multi-input, multi-output systems. In an example, the technique is applied to the despin control system of a dual-spin satellite.

INTRODUCTION

It is often desirable to design a compensator as a transfer function or a matrix of transfer functions, because the powerful and easy-to-use techniques of frequency domain shaping can be employed. On the other hand, it is often desirable to implement the compensator in estimator-controller form, in order to have estimates of the system states available. Having the state estimates available makes it easy to monitor the performance of the system if the state variables are meaningful system variables. It is well known that compensators can almost always be realized in an estimator-based form; this was shown, for example, in [1]. However, a technique for mapping an arbitrary compensator to estimator-controller form has not been available until now. In this paper a technique is derived for computing the estimator and controller gains to implement a given linear, time-invariant compensator in estimator-controller form. This is done by mapping the states of the compensator to those of the estimator-controller using a linear transformation, called a Lyapunov transformation in [1]. This transformation satisfies a nonsymmetric matrix Riccati equation. It also transforms the original feedback system to a form with a block triangular transition matrix whose states are the estimator error and the plant states. This is a special case of a block-triangularizing transformation satisfying a nonsymmetric matrix Riccati equation. A previous control-theoretic application of this transformation can be found in [2]. In the next section the technique is derived and its solution is discussed. Following that are some remarks on applications and extensions of the mapping technique. As an example, this technique is then applied to the platform despin control system of a dual-spin satellite. In Appendix A the results for various types of sampled-data systems are presented. In Appendix B an alternative interpretation of the results and the relationship to the results in [2] are presented. The results are summarized for the various types of estimators in Table 1.

DERIVATION

In this section it will be shown how to find the estimator-controller form of a compensator for the simplest possible case, a continuous-time, strictly proper compensator and plant model.

The derivation proceeds as follows. The plant is defined by:

$$\dot{x} = Fx + Gu$$
$$y = Hx.$$

A feedback compensator (assumed given) for this plant is described in state space form by the equations:

$$\dot{z} = Az + By \tag{1}$$
$$u = Cz. \tag{2}$$

This compensator will be mapped to the full-order estimator-controller described by the equations:

$$\dot{\hat{x}} = F\hat{x} + Gu + K_o(y-H\hat{x})$$
$$u = -K_c x$$

or $$\dot{\hat{x}} = (F - GK_c - K_oH)\hat{x} + K_oy \tag{3}$$

In order to map the compensator to this estimator-based form, the state vectors of the compensator and the estimator-controller are related by a linear transformation:

$$\hat{x} = Tz. \tag{4}$$

Substituting into the compensator equations (1) and (2), we get:

$$x = Tz = TAz + TBy = TAT^{-1}\hat{x} + TBy$$
$$u = Cz = CT^{-1}\hat{x}.$$

This leads to the relationships (using (3) and (4)):

$$TAT^{-1} = F - GK_c - K_oH \tag{5}$$
$$K_o = TB \tag{6}$$
$$K_c = -CT^{-1} \tag{7}$$

Substituting (6) and (7) into (5), and postmultiplying by T, we get the nonsymmetric algebraic Riccati equation,

$$TBHT + TA - FT - GC = 0 \tag{8}$$

Any real invertible solution T can be used to compute the estimator and controller gains K_o and K_c using equations (6) and (7). The existence and invertibility of T are equivalent to the existence of an estimator-based version of the original compensator defined by the matrices (A,B,C) and equations (1) and (2). A compensator always has an estimator-based realization as long as it is not necessary to split complex conjugate pairs of closed loop poles between the closed loop estimator poles (eigenvalues of $F-K_oH$) and the closed loop controller poles (eigenvalues of $F-GK_c$) ([7]).

The closed-loop system poles will be the eigenvalues of the closed-loop system matrix

$$M = \begin{vmatrix} A & BH \\ GC & F \end{vmatrix} .$$

The closed-loop system poles (eigenvalues of M) are the union of the closed-loop estimator poles and the closed-loop controller poles.

The Riccati equation can be solved using the Hamiltonian eigenvector approach of Potter [3]. An eigenvector decomposition of the system matrix is done, to get:

$$U^{-1}MU = \Lambda .$$

If none of the closed-loop poles with the compensator specified are repeated (M has no repeated eigenvalues), the columns of U are the eigenvectors of M and Λ is a diagonal matrix whose elements are the eigenvalues of M. U is then partitioned as

$$U = \begin{vmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{vmatrix}$$

Solutions of the Riccati equation can be computed as $T = U_{21}U_{11}^{-1}$. The various solutions of the Riccati equation will correspond to the possible orderings of the columns of U. There can be as many as $(2n)!/(n!)^2$ of these solutions. See [4] for more detail on the solutions of the nonsymmetric Riccati equation. If M has any repeated eigenvalues, it cannot be diagonalized and a Jordan form decomposition must be done. The details for this case are in [4]. Instead of a diagonal or Jordan form, M can be transformed to an upper triangular form; the columns of U in this case are called Schur vectors. This approach requires less computation and is numerically more stable [5].

Remarks

If the order of the plant is n, the eigenvalues of M corresponding to the first n eigenvectors of U are the closed loop controller poles. The remaining eigenvalues of M are the closed loop estimator poles. If the eigenvectors are ordered so that the eigenvalues of M corresponding to the first n eigenvectors are self-conjugate, the corresponding solution of the Riccati equation will be real. If a complex conjugate pair of eigenvalues is split between the estimator and controller, the corresponding gain matrices K_O and K_C and the Riccati solution T will be complex. Uncontrollable plant poles must be eigenvalues of $(F-GK_C)$, and unobservable plant poles must be eigenvalues of $(F-K_OH)$. If either condition is violated when ordering the eigenvectors of the Hamiltonian matrix, the corresponding Riccati solution T will be singular.

If any invertible real solution of the Riccati equation exists (and thus if any set of controller and estimator gains exist to implement the estimator-controller) at least two will exist. This corresponds to exchanging a pair of closed loop system poles between the estimator and the controller. Some considerations for dividing the closed loop poles between the estimator and the controller are discussed in [6]. The number of solutions can be large (as high as 70 for a fourth order compensator). On the other hand, for a second-order system either zero (if both closed loop poles are complex) or two solutions exist.

The analysis can be extended to sampled data systems, or systems with continuous propagation and a sampled data sensor (with the state estimate used to compute the control being the a priori or the a

posteriori estimate). This is discussed in Appendix A.

The results for the simple system derived above and the extensions of Appendix A are summarized in Table 1.

This technique can be used with a reduced-order estimator. This will be discussed in a future paper.

The transformation described in this paper is a special case of a well-known class of transformations for block-triangularizing or block-diagonalizing matrices. A previous application to control theory was done by Kokotovic ([2]). This is discussed in Appendix B.

EXAMPLE

In the following example the technique is applied to an estimator-based platform despin compensator for a dual-spin satellite. The plant is a double-integrator with transfer function

$$\frac{y(S)}{u(S)} = \frac{1}{IS^2} .$$

The control input u is the control torque in N-m and the output y is the platform phase in radians. The control objective is to regulate y to zero. I is the platform spin-axis moment of inertia, 1084.7 kg-m^2.

A constant bias torque adding in with the control torque is modelled as a third state. This adds an uncontrollable mode at the origin in the S-plane to the plant dynamics. Thus, $F-GK_c$ always has an eigenvalue at zero.

The compensator used on the spacecraft is actually sampled-data, using sampled measurements. What is done in practice is to design the compensator as a Laplace transfer function using frequency domain techniques such as Bode and Nichols charts. A state space realization of the compensator is then derived, and this is mapped to a sampled-data compensator using (for example) the bilinear transform. For simplicity we will only present the continuous version here.

The plant is described in state space form by:

$$F = \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1/I \\ 0 & 0 & 0 \end{vmatrix} \qquad G = \begin{vmatrix} 0 \\ 1/I \\ 0 \end{vmatrix}$$

$$H = \begin{vmatrix} 1 & 0 & 0 \end{vmatrix}$$

where the plant states are, in order, the platform phase (radians) and rate (radians per second) and the bias torque (N-m). The transfer function of the compensator to be converted to estimator-controller form is:

$$\frac{u(S)}{y(S)} = \frac{-33895(S+.18)(S+.4)}{S(S^2+2(.5)(6)S+(6)^2)} \quad \text{N-m/rad}$$

A state space version of this compensator is:

$$A = \begin{vmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & -3.0 & 5.196 \\ 0.18 & -5.196 & -3.0 \end{vmatrix} \qquad B = \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix}$$

$$C = \begin{vmatrix} 0.0 & 16960 & -33895 \end{vmatrix} .$$

What is desired is the estimator gain vector

$$K_o = \begin{vmatrix} K_{o1} \\ K_{o2} \\ K_{o3} \end{vmatrix}$$

and the control gain vector

$$K_c = | K_{c1} \; K_{c2} \; K_{c3} \; |.$$

When the algorithm described above is carried out, it is found that the closed-loop system poles (the eigenvalues of M) are:

real poles at: $S = 0.0$ and complex pairs at: ($\zeta = 0.458$, $\omega_n = 5.546$)
$S = -0.161$ ($\zeta = 0.561$, $\omega_n = 0.675$).

There are two real poles and two pairs of complex conjugate poles. The real pole at 0 rad/sec is an uncontrollable mode and thus will always be an eigenvalue of the closed loop controller, $(F-GK_c)$. Then, because a complex conjugate pair of poles cannot be split between the estimator and the controller, the other real pole will be a pole of the estimator. One of the pairs of complex conjugate poles will be poles of the estimator and the other will be closed loop controller poles. For realizing the compensator, it is arbitrary which pair corresponds to the controller and which pair corresponds to the estimator; the compensator will have the same transfer function either way. Therefore, there are two possible estimator-based realizations of the compensator, and two real invertible solutions of the Riccati equation.

The two possible sets of estimator and controller gains are:

	Solution 1	Solution 2
estimator gains:		
K_{o1} = (rad/rad)	0.9183	5.242
K_{o2} = (rad/(rad/sec))	0.5772	31.57
K_{o3} = (N-m/rad)	79.34	5357.2
estimator poles (eigenvalues of $(F-K_oH)$):	$S = -.161$ and ($\zeta = .561$, $\omega_n = .675$)	$S = -.161$ and ($\zeta = .458$, $\omega_n = 5.546$)
control gains:		
K_{c1} = (N-m/rad)	33360.	494.1
K_{c2} = (N-m/(rad/sec))	5511.9	821.9
K_{c3} = (N-m/N-m)	1.0	1.0
controller poles (eigenvalues of $(F-GK_c)$):	$S = 0.0$ and ($\zeta = .458$, $\omega_n = 5.546$)	$S = 0.0$ ($\zeta = .561$, $\omega_n = .675$)

It was desired to place the faster poles in the estimator, so solution 2 was chosen. [6] gives more detail on how to divide the closed loop poles between the estimator and the controller.

CONCLUSIONS

The transformation between the states of an arbitrary single- or multivariable compensator and an equivalent estimator-controller

satisfies a nonsymmetric matrix Riccati equation. Any real nonsingular solution of this equation can be used to compute the estimator and controller gains to cause the estimator-controller to have the same transfer function as the specified compensator. For a given compensator transfer function (or equivalently, for a given set of closed-loop poles) there can be many sets of estimator and controller gains (or equivalently, several real invertible solutions of the corresponding Riccati equation). The solutions of the Riccati equation can be computed by doing an eigenvector decomposition of the closed-loop system matrix, and the solution is a transformation between two versions of the system matrix. It is well known that this type of block-triangularizing transformation satisfies a nonsymmetric matric Riccati equation. A future paper will contain more analytical background and the analysis for various types of reduced order estimators.

APPENDIX A. APPLICATION TO SAMPLED-DATA SYSTEMS

The technique discussed in this paper can be extended to sampled data systems with little modification. In this Appendix the results will be summarized for three types of sampled-data systems. The first is an arbitrary sampled data plant and compensator; for simplicity, we will analyze the case with no direct feedthrough in the plant or compensator. The other two cases are versions of estimators in predictor-corrector form. One version uses the a priori estimate to compute the control; here we allow the plant model but not the compensator to have a feedthrough. The other uses the a posteriori estimate to compute the control; in this case we allow the compensator but not the plant to have a direct feedthrough.

i) Simplest possible sampled compensator and plant; no feedthrough.

In this case, the plant is defined by

$$x_{k+1} = Fx_k + Gu_k$$
$$y_k = Hx_k.$$

The compensator is described by:

$$z_{k+1} = Az_k + By_k$$
$$u_k = Cz_k.$$

This compensator is be mapped to the estimator-controller described by the equations:

$$\hat{x}_{k+1} = F\hat{x}_k + Gu_k + K_o(y_k - H\hat{x}_k)$$
$$u_k = -K_c\hat{x}_k$$

This is exactly the same as the continuous case, with the differentiation operator replaced by the time advance operator. Thus, we get the same answer as before; namely, that one solves the nonsymmetric Riccati equation (8) for T and substitutes the result into equations (6) and (7) to compute K_o and K_c.

ii) Predictor-corrector form; a priori estimate is used for control computation; feedthrough allowed in plant but not compensator.

In this case and the following, the estimator is implemented as a predictor-corrector. This situation comes about, for example, with a

continuous plant and sampled measurements. The estimate after the prediction and before the correction is used for the control; that is, the control is computed as $u_k = -K_c\hat{x}_{k/(k-1)}$.

We allow the plant to have a feedthrough here; it is defined by:

$$x_{k+1} = Fx_k + Gu_k$$
$$y_k = Hx_k + Ju_k.$$

The compensator is described by:

$$z_{k+1} = Az_k + By_k$$
$$u_k = Cz_k.$$

The estimator is described by:

prediction (state propagation): $\hat{x}_{k/(k-1)} = F\hat{x}_{(k-1)/(k-1)} + Gu_k$

correction: $\hat{x}_{k/k} = \hat{x}_{k/(k-1)} + K_o[y_k - H\hat{x}_{k/(k-1)} - Ju_k]$

control: $u_k = -K_c\hat{x}_{k/(k-1)}.$

Combining the estimator equations (substituting the correction equation into the prediction equation and using $u_k = -K_c x_{k/k}$ for the control feedback into the predictor), one gets:

$$\hat{x}_{(k+1)/k} = (F-GK_c)(I-K_oH+K_oJK_c)\hat{x}_{k/(k-1)} + (F-GK_c)K_oy_k$$
$$u_k = -K_c\hat{x}_{k/(k-1)}$$

When this is mapped to the compensator using $\hat{x}_{k/(k-1)} = Tz_k$, and the appropriate substitutions made, one finds that the Riccati equation is:

$$TBHT + T(A+BJC) - FT - GC = 0,$$

and that the gains can be computed from:

$$TB = (F-GK_c)K_o \quad \text{and} \quad CT^{-1} = -K_c.$$

If $u_k = -K_c\hat{x}_{k/(k-1)}$ instead of $u_k = -K_c\hat{x}_{k/k}$ is used to compute the control feedback into the predictor, with no direct feedthrough in the plant model then the situation is the same as in case (i) above with K_o replaced by FK_o.

iii) Predictor-corrector form; a posteriori estimate is used for control computation; feedthrough allowed in compensator but not plant.

This is similar to case (ii), except that the estimate as computed at the correction is used for the control computation (that is, $u_k = -K_c\hat{x}_{k/k}$). The estimator correction equation is the same as before; the plant (prediction) equation is the same, except that no feedthrough (J) term is allowed. In this case one substitutes the prediction equation into the correction equation and gets:

$$\hat{x}_{k/k} = (I-K_oH)(F-GK_c)\hat{x}_{(k-1)/(k-1)} + K_oy_k$$
$$u_k = -K_c\hat{x}_{k/k}$$

This is not in state space form, because y_k is used to compute $\hat{x}_{k/k}$. The following state space system has the same transfer function:

$$\hat{x}_{(k+1)/k} = (F-GK_c)(I-K_oH)\hat{x}_{k/(k-1)} + (F-GK_c)K_oy_k$$
$$u_k = -K_c\hat{x}_{k/k} = -K_c(I-K_oH)\hat{x}_{k/(k-1)} - K_cK_oy_k.$$

This estimator-controller has a feedthrough (a special case of the analysis in Appendix A); thus, the compensator being mapped must also have a feedthrough. This happens often in sampled-data systems; for example, if the bilinear transform is used to derive a sampled-data compensator from a continuous one the sampled-data version will have equal order in the numerator and the denominator. The compensator is then described by:

$$z_{k+1} = Az_k + By_k$$
$$u_k = Cz_k + Dy_k.$$

The analysis proceeds by mapping the estimator-controller to the compensator using $\hat{x}_{k/(k-1)} = Tz_k$.

The Riccati equation for this system is:

$$TBHT + TA - (F+GDH)T - GC = 0.$$

The gains can be computed from:

$$K_c = -DH - CT^{-1}, \text{ then } K_o = (F+GCT^{-1}+GDH)TB = (F-GK_c)TB.$$

It can be shown that this estimator-controller has a zero at $Z = 0$, or equivalently, a free time advance (in each path, for multivariable systems). If one attempts to map a compensator which does not have a zero at $Z = 0$ in every path, the result will be incorrect. The compensator will have a zero at $Z = 0$ if $D = CA^{-1}B$. A^{-1} always exists for a sampled system except one with a time delay of at least one sample period. Because the Z-transfer function of the compensator must have a zero at $Z = 0$, the Laplace transform of the compensator is not arbitrary. For example, if the bilinear transformation with prewarping [9] is used to map from a continuous to a discrete compensator, the Laplace transfer function of the continuous compensator must have a zero at $S = -\pi/(2T)$ to give a Z-plane zero at $Z = 0$.

APPENDIX B. ALTERNATIVE INTERPRETATION OF THE TRANSFORMATION T

The original closed loop system made up of the plant and the desired compensator in the feedback loop is described, in the simplest case, by:

$$\begin{vmatrix} \dot{z} \\ \dot{x} \end{vmatrix} = \begin{vmatrix} A & BH \\ GC & F \end{vmatrix} \begin{vmatrix} z \\ x \end{vmatrix}.$$

Note that the system matrix is the same as the matrix M for the Riccati equation that must be solved in the mapping described in the main body of the report.

The closed loop system made up of the plant with an estimator-controller in the feedback path is described by:

$$\begin{vmatrix} \dot{e} \\ \dot{x} \end{vmatrix} = \begin{vmatrix} F-K_oH & 0 \\ GK_c & F-GK_c \end{vmatrix} \begin{vmatrix} e \\ x \end{vmatrix}$$

where $e = x - \hat{x}$.
The transformation between these two systems, using $\hat{x} = Tz$ as before, is:

$$\begin{vmatrix} e \\ x \end{vmatrix} = \begin{vmatrix} -T & I \\ 0 & I \end{vmatrix} \begin{vmatrix} z \\ x \end{vmatrix}.$$

This gives a similarity transformation between the state transformation matrices of the compensator-feedback system and the system with an estimator-controller in the feedback:

$$\begin{vmatrix} F-K_oH & 0 \\ GK_c & F-GK_c \end{vmatrix} = N \begin{vmatrix} A & BH \\ GC & F \end{vmatrix} N^{-1},$$

$$\text{with } N \equiv \begin{vmatrix} -T & I \\ 0 & I \end{vmatrix}.$$

Thus, T is part of a block-triangularizing transformation. This is related to the result of Kokotovic ([2]), in which it is shown that the block-triangularizing transformation for an arbitrary system satisfies a nonsymmetric Riccati equation.

REFERENCES

[1] O'Reilly, J., Observers for Linear Systems. London: Academic Press, 1983.

[2] Kokotovic. P. V., "A Riccati Equation for Block-Diagonalization of Ill-Conditioned Systems," IEEE Trans. Auto. Control, vol. AC-20, No. 6, Dec., 1975, pp. 812-814.

[3] Potter, J. E., "Matrix Quadratic Solution," SIAM J. Applied Math., vol. 14, 1966, pp. 496-501.

[4] Medanic, J., "Geometric Properties and Invariant Manifolds of the Riccati Equation," IEEE Trans. Auto. Control, vol. AC-27, No. 3, June, 1982, pp. 670-676.

[5] Laub, A. J., "A Schur Method for Solving Algebraic Riccati Equations," IEEE Trans. Auto. Control, vol. AC-24, no. 2, April, 1979.

[6] Bender, D. J., "Considerations for Estimator-Based Control Systems," to appear in International Journal of Control, 1984

[7] Schumacher, J. M., "Compensator Synthesis Using (C,A,B)-Pairs," IEEE Trans. Auto. Control, vol. AC-25, No. 6, Dec., 1980, pp. 1133-1138.

[8] Moore, B. C., "On the Flexibility Offered by State Feedback in Multivariable Systems Beyond Closed Loop Eigenvalue Assignment," IEEE Trans. Auto. Control, vol. AC-21, no. 5, Oct., 1976, pp. 689-692.

[9] Franklin, G. F. and J. D. Powell, Digital Control of Dynamic Systems. Reading, Mass: Addison-Wesley,1981

TABLE 1: MAPPING COMPENSATORS TO ESTIMATOR/CONTROLLER FORM: A SUMMARY

	CONTINUOUS OR PURE SAMPLED SYSTEM; NO FEEDTHROUGH IN PLANT OR COMPENSATOR	SAMPLED SYSTEM: A PRIORI ESTIMATE USED FOR CONTROL (FEEDTHROUGH IN PLANT, NO FEEDTHROUGH IN COMPENSATOR)	SAMPLED SYSTEM: A POSTERIORI ESTIMATE USED FOR CONTROL (FEEDTHROUGH IN COMPENSATOR, NO FEEDTHROUGH IN PLANT)	CONTINUOUS OR PURE SAMPLED SYSTEM WITH FEEDTHROUGH IN PLANT AND COMPENSATOR
PLANT EQUATION OF MOTION	$\begin{cases} \dot{x} = Fx + Gu \\ y = Hx \end{cases}$ OR $\begin{cases} x_{k+1} = Fx_k + Gu_k \\ y_k = Hx_k \end{cases}$	$\begin{aligned} x_{k+1} &= Fx_k + Gu_k \\ y_k &= Hx_k + Ju_k \end{aligned}$	$\begin{aligned} x_{k+1} &= Fx_k + Gu_k \\ y_k &= Hx_k \end{aligned}$	$\begin{cases} \dot{x} = Fx + Gu \\ y = Hx + Ju \end{cases}$ OR $\begin{cases} x_{k+1} = Fx_k + Gu_k \\ y_k = Hx_k + Ju_k \end{cases}$
COMPENSATOR	$\begin{cases} \dot{z} = Az + By \\ u = Cz \end{cases}$ OR $\begin{cases} z_{k+1} = Az_k + By_k \\ u_k = Cz_k \end{cases}$	$\begin{aligned} z_{k+1} &= Az_k + By_k \\ u_k &= Cz_k \end{aligned}$	$\begin{aligned} z_{k+1} &= Az_k + By_k \\ u_k &= Cz_k + Dy_k \end{aligned}$	$\begin{cases} \dot{z} = Az + By \\ u = Cz + Dy \end{cases}$ OR $\begin{cases} z_{k+1} = Az_k + By_k \\ u_k = Cz_k + Dy_k \end{cases}$
ESTIMATOR/ CONTROLLER	$\begin{cases} \dot{\hat{x}} = F\hat{x} + Gu + K_o(y - H\hat{x}) \\ u = -K_C\hat{x} \end{cases}$ OR $\begin{cases} \hat{x}_{k+1} = F\hat{x}_k + Gu_k + K_o(y_k - H\hat{x}_k) \\ u_k = -K_C\hat{x}_k \end{cases}$	$\begin{cases} \hat{x}_{k+1/k} = F\hat{x}_{k/k} + Gu_k \\ \hat{y}_{k+1/k} = H\hat{x}_{k+1/k} + Ju_{k+1} \\ \hat{x}_{k/k} = \hat{x}_{k/k-1} + K_o(y_k - \hat{y}_{k/k-1}) \\ u_k = -K_C\hat{x}_{k/k-1} \end{cases}$ $\Downarrow$ $\begin{cases} \hat{x}_{k+1/k} = (F - GK_C)(I - K_oH + K_oJK_C)\hat{x}_{k/k-1} \\ \quad + (F - GK_C)K_oy_k \\ u_k = -K_C\hat{x}_{k/k-1} \end{cases}$	$\begin{cases} \hat{x}_{k+1/k} = F\hat{x}_{k/k} + Gu_k \\ \hat{y}_{k+1/k} = H\hat{x}_{k+1/k} \\ \hat{x}_{k/k} = \hat{x}_{k/k-1} + K_o(y_k - \hat{y}_{k/k-1}) \\ u_k = -K_C\hat{x}_{k/k} \end{cases}$ $\Downarrow$ $\begin{cases} \hat{x}_{(k+1)/(k+1)} = (I - K_oH)(F - GK_C)\hat{x}_{k/k} + K_oy_k \\ u_k = -K_C\hat{x}_{k/k} \end{cases}$ OR $\begin{cases} \hat{x}_{k/k-1} = (F - GK_C)(I - K_oH)\hat{x}_{k/k-1} + (F - GK_C)K_oy_k \\ u_k = -K_C[(I - K_oH)\hat{x}_{k/k-1} + K_oy_k] \end{cases}$ USING $\hat{x}_{k/k} = (I - K_oH)\hat{x}_{k/k-1} + K_oy_k$	$\begin{cases} \dot{\hat{x}} = F\hat{x} + Gu + K_o(y - \hat{y}) \\ u = -K_C\hat{x} + Dy \\ \hat{y} = H\hat{x} + Ju \end{cases}$ $\Downarrow$ $\begin{cases} \dot{\hat{x}} = (F - GK_C - K_oH + K_oJK_C)\hat{x} + (GD + K_o - K_oJD)y \\ u = -K_C\hat{x} + Dy \end{cases}$ OR $\begin{cases} \hat{x}_{k+1} = F\hat{x}_k + Gu_k + K_o(y_k - \hat{y}_k) \\ u_k = -K_C\hat{x}_k + Dy_k \\ \hat{y}_k = H\hat{x}_k + Ju_k \end{cases}$ $\Downarrow$ $\begin{cases} \hat{x}_{k+1} = (F - GK_C - K_oH + K_oJD)\hat{x}_k + (GD + K_o - K_oJD)y_k \\ u_k = -K_C\hat{x}_k + Dy_k \end{cases}$
TRANSFORMATION USED	$\hat{x} = Tz$	$\hat{x}_{k/k-1} = Tz_k$	$\hat{x}_{k/k-1} = Tz_k$	$\hat{x} = Tz$
RICCATI EQUATION	$TBHT + TA - FT - GC = 0$	$TBHT + T(A + BJC) - FT - GC = 0$	$TBHT + TA - (F + GDH)T - GC = 0$	$TB(I - JD)^{-1}HT + T[A + B(I - JD)^{-1}JC] - [F + GD(I - JD)^{-1}H]T - GC - GD(I - JD)^{-1}JC = 0$
GAINS ARE COMPUTED FROM:	$\begin{aligned} K_o &= TB \\ K_C &= CT^{-1} \end{aligned}$	$\begin{aligned} K_o &= (F - GK_C)^{-1}TB \\ K_C &= -CT^{-1} \end{aligned}$	$\begin{aligned} K_o &= (F + GCT^{-1} + GDH)TB = (F - GK_C)TB \\ K_C &= -DH - CT^{-1} \end{aligned}$	$\begin{aligned} K_o &= (TB - GD)(I - JD)^{-1} \\ K_C &= CT^{-1} \end{aligned}$
CLOSED-LOOP SYSTEM MATRIX IS:	$\begin{bmatrix} A & BH \\ GC & F \end{bmatrix}$	$\begin{bmatrix} A + BJC & BH \\ GC & F \end{bmatrix}$	$\begin{bmatrix} A & BH \\ GC & F + GDH \end{bmatrix}$	$\begin{bmatrix} A + B(I - JD)^{-1}JC & B(I - JD)^{-1}H \\ GC + GD(I - JD)^{-1}JC & F + GD(I - JD)^{-1}H \end{bmatrix}$

839851 11

DESIGN OF MINIMAL MULTIFUNCTIONAL OBSERVER

B. SHAFAI & R.L. CARROLL
George Washington University
Department of EE & CS
Washington, D.C. 20052

ABSTRACT

In this paper we present a simple method for the design of minimal order observer capable of reconstructing linear functions of the states of a linear system. The aim of the method is to solve the fixed pole and the free pole observer problems by systematically increasing the order until all observer requirements are satisfied. The necessary and sufficient condition for constructing the observer is derived based on state space in an algebraic setting and the design method is presented by means of numerical examples. Direct procedure is also given for the design of single functional observer.

I. INTRODUCTION

Since Luenberger's original works on Observer Theory [1] - [3], there has been numerous publications on the subject. The state observer problem is solved completely whereas the functional observer problem remains as an incompletely solved problem.

In recent years the design of stable minimal order functional observer attracted many researchers. Unfortunately, satisfactory design procedure has not been established simply because of lack of knowledge in structural properties of functional observers.

For an early treatment of the problem we refer to the works of Fortman and Williamson [4], Murdoch [5], [6], Moore and Ledwich [7], partial realization approach of Roman and Bullock [8], frequency domain approach of Wang and Davison [9], and its extension by Wolovich et. al. [10]. One may also see the references cited in these papers.

All of the above design techniques seem to possess undesirable features. The complexity is often accompanied by loss of design freedoms such as, stability vs. minimality, assignability of observer eigenvalues vs. system parameters and design algorithms that are parameter dependent. Recent works of Kimura [11],[12], Furuta and Kawaji [13],[14], based on geometric approach of Wonham [15], [16] gave new insight on structural properties and better understanding of the problem.

As a result further contributions have been made in this area. Sirisena [17] proposed a systematic procedure for characterizing candidate observers of minimal order that satisfies all the requirements except stability. Kajiwara et. al. [18] provided more insight into the structure of functional observer problem and the fine work of Fairman and Gupta [19] yield new and interesting results parallel to the work of Kimura [12]. However, this method requires a computationally undesirable transformation to the Luenberger observable canonical form and evaluation of structural indices. Moreover, it seems that the approach leads to observer with order greater than may be necessary once the necessary condition of Theorem 1 in their paper is not satisfied.

II. PROBLEM FORMULATION AND PRELIMINARIES

In order to formulate the problem more precisely, consider a linear system described by

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx \tag{2}$$

where $x \in R^n$, $u \in R^m$ and $y \in R^p$ are state, input and output vectors respectively, and the matrices A, B and C are of appropriate dimensions. Our problem is to construct an observer of the form

$$\dot{z} = Fz + Gy + Hu \tag{3}$$

$$\hat{w} = Dz + Ey \tag{4}$$

for estimating linear functions of the states represented by

$$w = Kx \tag{5}$$

where $w \in R^q$ and $z \in R^r$ and $\hat{w}$ is an asymptotic estimate of w in the following sense,

$$\lim_{t\to\infty} \frac{d^i}{dt^i} (\hat{w}-w) = 0_q \tag{6}$$

The observer is required to be stable and the order r being as small as possible.

This problem often arises in optimal control or pole placement problems when we are to implement the control law via incomplete state observations. In this case w represents u and q = m.

There are two distinct observer problems which may be termed as the fixed-pole stable observer problem and the free-pole stable observer problem. In the former case the eigenvalues of the observer are arbitrary prespecified, whereas in the later case the eigenvalues are required to lie in the open left half of s-plane and they will be postspecified.

Structurally we may also distinguish two types of observer problems, these are known as Kalman type and Luenberger type problems. An observer is of Kalman type if there is no direct output feedthrough, i.e. E = 0 otherwise it is of Luenberger type.

Our aim is to give the simplest possible design procedure which satisfies both requirements of stability and minimality. We consider the solution of both the fixed-and free-pole problem for Luenberger type observer. However, it should be noted that the scheme can also be adopted for the Kalman type observer by simple manipulation. The transformation to Luenberger canonical form is not required in the present approach. The method starts with an observer having the least possible order satisfying a certain condition. If this fails the order of the observer is systematically increased until the condition is satisfied.

III. DEVELOPMENT OF THE METHOD

It is well known that the system (3),(4) is a minimal order observer for the system (1),(2) if and only if there exists an r x n matrix T relating the observer states z and the system states x by a linear equation of the form

$$z = Tx + e \tag{7}$$

such that the following conditions hold for a specified qxn matrix k

$$TA-FT = GC \tag{8}$$

$$TB = H \tag{9}$$

$$DT+EC = K \tag{10}$$

$$F \text{ is stable i.e. Re } \lambda_i(F)<0 \ \forall i \tag{11}$$

$\lambda_i(F)$, $\forall i$ are prespecified in the fixed-pole problem. Once (8)-(11) are satisfied by appropriate choice of observer parameters, the observer state z approaches Tx asymptotically which implies that $\hat{w}$ is an asymptotic estimate of w.

Without loss of generality we assume that the system is observable and controllable and the rank of matrices C and K are full. Furthermore, we assume that the system (1),(2) is already transformed to the following form.

$$\dot{x} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} x + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u \qquad (1)'$$

$$y = [0_{px(n-p)} I_p]\, x \qquad (2)'$$

Where the partitioning of A and B are in accordance with the output equation and the submatrices are of appropriate dimensions. The transformation is obtained by arbitrary selecting an (n-p) x n matrix $\tilde{C}$ in such a way that $[\tilde{C}'\ C']'$ is a nonsingular matrix. Equation (10) implies that the columns of K' are contained in the column span of [T' C']. If K = I then (3),(4) is a minimal state observer and the structure of the matrix T is specified by

$$T = [I_r \quad L]$$

Where r = n-p.

Suppose n>r>n-p, then it is easy to show that the structure of the matrix T is specified by

$$T = \left[\begin{array}{c|c} I_{n-p} & L_1 \\ \hline 0 & L_2 \end{array}\right]$$

and for the case of r = n the matrix T becomes identity.

This motivated us to define a structure for the matrix T when r<n-p.

Theorem 1. The necessary condition for a minimal order observer to exist is that the matrix T to be of full rank.

Proof. Let us denote the system and the observer states as

$$x_c = \begin{bmatrix} x \\ z \end{bmatrix}$$

Then the composite system

$$\dot{x}_c = \begin{bmatrix} A & 0 \\ GC & F \end{bmatrix} x_c + \begin{bmatrix} B \\ H \end{bmatrix} u$$

must be controllable.

Using (8) and (9) the controllability matrix of the composite system can be written as follows

$$C_c = \left[\begin{array}{c|c|c|c} B & AB & \cdots & A^{n+r-1}B \\ \hline TB & TAB & \cdots & TA^{n+r-1}B \end{array}\right]$$

Since the original system is controllable, the first n rows of C_c are linearly independent. Obviously T must be of full rank in order to guarantee the nonsingularity of C_c . Q.E.D.

Definition 1. A functional observer is called feasible if $\rho[T'C']' = r+p$, where ρ denotes the rank of a matrix.

Partitioning K and T conformally with C, enables one to write (8) and (10) in the following form.

$$T_1 A_{11} + T_2 A_{21} = FT_1 \qquad (12)$$

$$T_1 A_{12} + T_2 A_{22} = FT_2 + G \qquad (13)$$

$$K_1 = DT_1 \qquad (14)$$

$$E = K_2 - DT_2 \qquad (15)$$

lemma 1. The necessary condition for a feasible functional observer of minimal order to exist is that the matrix T_1 to be of full rank.

Proof. The proof of the above lemma is a direct consequence of the definition 1 and the Theorem 1.

In order to guarantee the feasibility, there are $\binom{n-p}{r}$ distinct possibilities to place all columns of I_r in matrix T_1. However, once an appropriate T_1 is selected, by appropriate rearrangement of the states the structure of the matrix T_1 can be specified as

$$T_1 = [I_r \; L] \tag{16}$$

where L is r x(n-p-r) submatrix.

It should be noted that by doing this rearrangement the form of the output matrix C does not change.

Next partition K_1 as

$$K_1 = [K_{11} K_{12}] \tag{17}$$

where K_{11} is q x r and K_{12} is q x (n - p - r) submatrices and by substituting (16) and (17) into (14), we get

$$K_{11} = D \tag{18}$$

$$K_{12} = D\,L \tag{19}$$

(18) and (19) yield the following chain relation

$$K_{12} = K_{11}\,L \tag{20}$$

with the general solution given by

$$L = K_{11}'(K_{11}K_{11}')^{-1}\,K_{12} + [I - K_{11}'(K_{11}K_{11}')^{-1}\,K_{11}]\,M \tag{21}$$

where M is an arbitrary constant matrix. It is always possible to choose T_1 such that (20) has a solution, i.e., $K_{12} \in R(K_{11})$. Coordinate Transformation could be applied to the pair $(A,[K' C']')$ to eliminate possible confusion for selecting T_1 and to reduce the labor involved in finding the matrix L.

Having obtained T_1, the equation (12) can be written as

$$T_2 A_{21} = F T_1 - T_1 A_{11} \tag{22}$$

and we can state the main theorem of this paper.

Theorem 2. A feasible functional observer of minimal order exists for the system (1), (2) if and only if the following consistency condition is satisfied

$$[FT_1 - T_1 A_{11}] \left\{ I - A_{21}^g A_{21} \right\} = 0 \tag{23}$$

where g stands for generalized inverse of a matrix.

Proof. The proof of this theorem is constructive. If the condition (23) holds then the general solution of (22) is

$$T_2 = \left[FT_1 - T_1 A_{11} \right] A_{21}^g + N \left[I - A_{21} A_{21}^g \right] \tag{24}$$

where N is an aribtrary matrix having the same dimension as T_2.

Conversely if an observer exists, then T_2 satisfies (22) and by applying the property $A_{21} = A_{21} A_{21}^g A_{21}$ the consistency condition (23) follows. Q.E.D.

The condition (23) may be satisfied for any F, or for a restricted set of F, or even a unique F. The procedure fails if (23) is not satisfied by any F. In the later case the observer order is increased systematically until a satisfactory design is achieved.

a. Design Procedures:

There are several ways to find a stable matrix F which satisfies (23). For example let the rx(n-p-r) matrix L be represented as a member of the following set:

$$\mathcal{L} = \left\{ L \mid LK_{12} = K_{11} \right\} \tag{25}$$

with the general solution of L given by the equation (21).

By different L identify the problem through the conditon equation (23). If the problem has a unique (L,F) solution, use the extended results of Fortman and Williamson to design the observer. In this case the choice of M is essential and can not be chosen arbitrarily. Otherwise, substitute $T_1 = [I_r L]$ with L given by (21) into the condition equation (23). Appropriately define the stable matrix F having prescribed set of eigenvalues and solve (23) for unknown parameter M and possibly unspecified elements of F for the fixed pole problem or in the case of free pole problem determine the stable matrix F by adjusting the free matrix M. Any available technique can be used to achieve this goal. These points are further clarified in numerical examples and design simplifications are discussed below. When the set $\mathcal{L}$ is used to identify the problem, several cases might occur.

Consider the case where F is always fixed by different choice of L. This unique F might be either stable or unstable. We claim that for unstable F a modification of Fortman and Williamson leads to a stable observer design. (see example 2).

Now consider the case where some elements of f_{ij} are obtained to be fixed and the remaining ones are arbitrary. In this case the unrestricted assignment of eigenvalues to F can be treated as follows:

(i) Determine the remaining f_{ij} elements such that F is stable

(ii) choose L from (25) and specify T_1

(iii) finally, T_2 is obtained from (24) and the observer is specified.

It should be pointed out that for a fixed L the procedure simplifies considerably. We can further investigate the problem of unrestricted assignment of eigenvalues to F by substituting (16) into (12) and partitioning the matrices corresponding to the structure of the T_1. Then, two separate equations are obtained as:

$$T_1 A_{111} + T_2 A_{211} = F \tag{26}$$

$$T_1 A_{112} + T_2 A_{212} = FL \tag{27}$$

where A_{111}, A_{112} are (n-p)xr and (n-p)x(n-p-r) submatrices of A_{11}; A_{211}, A_{212} are pxr and px(n-p-r) submatrices of A_{21}, respectively. The problem is reduced to find T_2 and F such that (26) and (27) are satisfied and F is a stable matrix having prescribed set of eigenvalues.

lemma 2. The unrestricted assignment of eigenvalues to F is always possible if and only if

$$U \in R(V) \tag{28}$$

where $$U = \left[T_1 A_{112} - T_1 A_{111} L \right]'$$

and $$V = \left[A_{211} L - A_{212} \right]'$$

provided that fixed pole problem is identified from the consistency condition (23).

This lemma can be shown as follows. Eliminate F from (26), (27) which results to $V T_2' = U$ (29)

If (28) is satisfied then the general solution of (29) is given by

$$T'_2 = V'(VV')^{-1} U + [I - V'(VV')^{-1} V]Q \qquad (30)$$

where Q is an arbitrary matrix having the same dimension as T'_2.

Now the problem is to choose the matrix Q so as to position the eigenvalues of F at the desired locations. This is a standard pole placement problem which can be solved through the equations (26) and (30).

b. Lower bound on observer order

In the following lower bounds on the dimension of an observer for estimating Kx are given. Equation (14) implies that the columns of K'_1 are contained in the column span of T'_1, which means that $\rho(T'_1) \geq \rho(K'_1)$. Therefore, the dimension of an observer which estimates Kx for the system (1), (2) is bounded below by the rank of K'_1. That is,

$$r \geq \rho(K'_1) \qquad (31)$$

Although stronger bounds are available such as those given by Roman and Bullock based on partial realization theory, they all require computation of structural indices and rank evaluation of Hankel matrices.

Since in the present approach we start with the lowest possible order and increase it successively by one, it is sufficient to give a simpler bound. Using the defined structure matrix T and applying the range space condition $K' \varepsilon R[T'C']$ we obtain r(n+q) + qp unknowns and qn + r(n-p) + r equations for the fixed pole problem which results to

$$r \geq \frac{q(n-p)}{q + p-1} \qquad (32)$$

and in the case of free pole problem we have qn+r(n-p) equations which result to

$$r \geq \frac{q(n-p)}{q+p} \qquad (33)$$

Note that the fraction in each case must be rounded up to an integer and the strongest bound is chosen for initial setting of observer order.

IV. DESIGN OF SINGLE FUNCTIONAL OBSERVER

In this section, we consider the solution of the functional observer for the case of q=1. The results obtained in the previous section can be applied directly to this case.

Let the observability indices of the pair (A,C) be denoted by $\{\nu_i; i = 1,2, \ldots, p\}$ then due to Gopinath's work [20] the pair (A_{11}, A_{21}) is observable having observability indices given by $\{\nu_i-1; i=1,2,\ldots,\rho\}$. Thus if the observability index of (A,C) is ν then the observability index of (A_{11}, A_{21}) is ν-1.

First, we show that it is always possible to design a functional observer of order ν-1 having arbitrary eigenvalues. The following theorem was originally given by Luenberger.

Theorem 3. A functional observer of order ν-1 with arbitrary eigenvalues can be constructed for the system (1),(2).

Proof. The proof of the Theorem follows directly from the observability properties discussed above. In order to show this for the approach presented here, we refer to the equation (29), which has a solution if and only if (28) is satisfied or correspondingly

$$n-p-r \leq p \qquad (34)$$

must hold. But for $r = \nu-1$ (34) can be written as

$$n-p + 1-\nu \leq p \qquad (35)$$

It is well known that the observability index ν is bounded by

$$n/p \leq \nu \leq n-p+1 \qquad (36)$$

Substituting the upper bound of ν from (36) into (35) we get $p>0$, which is obviously true. Therefore, functional observer of order $\nu-1$ can be designed without restriction of observer eigenvalues. Q.E.D.

Gupta et. al. [21] have shown that the smallest possible order for an observer capable of reconstructing W is μ_o where $\{\mu_i; i=0,1...,p\}$ are structural indices of the pair $(A_{11}, [K_{11}' A_{21}']')$. But this time, with respect to the present method, the condition (28) must be checked in order to be able to design observers having dimension in $[\mu_o, \nu-2]$ with eigenvalues remaining arbitrarily assignable.

Design Algorithm:

We are now in a position to give a procedure for finding a $\nu-1$ order observer for estimating a single functional of the states. The algorithm is as follows:

Step 0. Transform the original system (A_o, B_o, C_o) and K_o to an algebraic equivalent system (A,B,C) and K defined by (1)', (2)'and (5).

Step 1. Obtain ν and define $T_1 = [I_{\nu-1} \; L]$ and D.

Step 2. Find the solution of $K_{12} = K_{11} L$ and specify T_1.

Step 3. Obtain the most general solution of T_2 given by (30) which may be expressed in the form

$$T_2' = R + SQ \qquad (37)$$

Step 4. Construct the following matrix equation by substituting (37) into (26).

$$(A_{111} T_1' + A_{211}' R) + (A_{211}' S) Q = F' \qquad (38)$$

Step 5. Choose the matrix Q so as to position the eigenvalues of F'at the desired locations. This is clearly a pole placement problem and standard techniques can be used to solve (38).

Step 6. Complete the observer design by evaluating the remaining parameters from (37), (9), (13) and (15).

V. Examples

The following examples illustrate different cases which might occur in functional observer problems. Some previously appeared examples are also discussed and solved using the present approach.

Example 1.

Let the problem parameters be given by

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 2 & -1 & -3 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}$$

$$C = [\,0 \quad 0 \quad 1\,] \qquad K = [\,5 \quad -7 \quad 1\,]$$

Since $\rho(k_1) = 1$, we set $r = 1$. Define (16), (18) as $T_1 = [1 \;\; L]$, $D = 5$ and obtain the unique solution of (20) as $L = -7/5$. $A^g{}_{21} = [1 \; 1]'$ yields $f=-23/9$ and the remaining observer matrices are determined from (24), (13), (15) and (9) as

$T_2 = -7/9$, $G = 28/81$, $E=41/9$, $H=[-53/45 \;\; 7/9]$

In this example we obtained a unique (L,F) solution. The famous example of Fortman and Williamson [4] belongs also to this category.

Example 2

Consider the example 3 in Sirisena [17] where

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 & -2 \\ 0 & 0 & 1 & 0 & -3.5 \\ 0 & 0 & 0 & 1 & -1.5 \end{bmatrix} \qquad B = \begin{bmatrix} -3 & 1 \\ 5 & 1 \\ -3 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & I \\ 2\times 3 & 2 \end{bmatrix} \qquad K = \begin{bmatrix} -1 & 1 & 0 & -1 & 2 \\ -1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

set r = 2 and specify T_1 and D from (16), (18) as

$$T_1 = \begin{bmatrix} 1 & 0 & \lambda_{11} \\ 0 & 1 & \lambda_{21} \end{bmatrix} \qquad D = K_{11} = \begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix}$$

since K_{11} is square nonsingular, obtain the unique solution of (20) from (21) as $L=[\lambda_{11}\ \lambda_{21}]' = [-1\ -1]'$ and with $A_{21}^{g} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ determine F from (23) as

$$F = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix}$$

which is the unique solution obtained by Sirisena. However, if we change the first column of the K matrix to $[1\ \ 1]'$ then $f_{12} = 1$ and we obtain an unstable solution for F by applying the steps presented in Sirensena. As we discussed previously, a modification of Fortman and Williamson [4] leads to the following stable solution.

$$\dot{z} = \begin{bmatrix} 0 & -0.42 \\ 1 & -0.5 \end{bmatrix} z + \begin{bmatrix} 0 & 0.12 \\ 0 & -0.77 \end{bmatrix} y + \begin{bmatrix} -1.52 \\ 6.34 \end{bmatrix} u$$

$$\hat{w} = [0.83\ \ -0.08]\ z + [0\ \ 0.29]\ y$$

Example 3

Consider the system given in Kawaji [14] and its associate gain matrix K as

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 3 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & I \\ 2\times 4 & 2 \end{bmatrix} \qquad K = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 & 2 & 0 \end{bmatrix}$$

It is not possible to construct a stable observer for r = 2. We set r = 3 and specify T_1 and D from (16), (18) as

$$T_1 = \begin{bmatrix} I & L \\ 3 & \end{bmatrix} \qquad D = K_{11} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

using the set (25), it follows that a three dimensional fixed pole observer can be constructed. Choose a solution for L from (21) as $L=[-1\ \ 0\ \ 1]'$ and solve (29) with solve (29) with $U = [2\ \ -1\ \ 0]$, $V = [-2\ \ 0]$ as follows: obtain the general solution of T_2 from (30) and substitute it in (26), we get

$$\begin{bmatrix} 0 & -1/2 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} Q = F'$$

Next find Q so as to position the eigenvalues of F at the desired locations specified by $s^3 + 6s^2 + 11s + 6$. Using any standard pole placement technique we obtain Q, F and automatically T_2 as

$$Q = \begin{bmatrix} 1 & 1 & 1 \\ 22 & -6 & -13 \end{bmatrix} \quad F = \begin{bmatrix} 0 & 22 & -1 \\ -1/2 & -6 & 0 \\ 0 & -12 & 0 \end{bmatrix} \quad T = \begin{bmatrix} -1 & 22 \\ 1/2 & -6 \\ 2 & 0 \end{bmatrix} \quad \begin{bmatrix} -13 \end{bmatrix}$$

and the remaining observer parameters can be determined by simple substitution.

Unlike the solution given by Kawaji [14] with one pole fixed at -1 and the remaining poles arbitrary, we provide a set of solutions with all poles arbitrary assignable. In fact, choose $L = [-2 \quad 0 \quad 2]'$ and by using a similar procedure we obtain a different stable solution for F.

Example 4

Consider the example in Kimura [12] which was reconsidered by Fairman et. al. [19]. In the design coordinate the parameters are

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 & -1 \\ -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0_{2x3} & I_2 \end{bmatrix} \qquad K = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 \end{bmatrix}$$

Set $r = 2$ and specify T_1 and D from (16), (18) as $T_1 = [I_2$ $L]$, $D = K_{11} = I_2$. A unique solution is obtained for L from (21) as $L = K_{12} = [0 \quad -1]'$.

From the consistency condition (23) we obtain $f_{11}=0$, $f_{21}=1$ and the remaining f_{1j} can be chosen arbitrarily. Therefore, a fixed pole observer of order 2 can be constructed with the desired eigenvalues specified by $s^2 + 3s + 2$. The stable matrix F and the matrix T_1 are determined as

$$F = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix} \qquad T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

and using $A_{21}^{g} = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$ the matrix T_2 can be obtained

as $T_2 = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$

The remaining observer parameters are obtained from (13), (15) and (9) by simple substitution. The result agrees with the one obtained in Fairman.

The famous example of Roman and Bullock [8] belongs also to this category and can be solved in a straight forward fashion.

Example 5

Consider the following system, which is taken fromn Gupta et.al.[21]

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C = [0_{3x4} \quad I_3] \qquad K = [1 \quad 2 \quad -1 \quad 0 \quad 1 \quad 2 \quad 1]$$

Since the observability index for this system is 3, it is possible to design an observer of order 2 with arbitrary eigenvalues specified by $s^2 + 3s + 2$.

Step 0. The system is already in the design coordinate.

Step 1. $\nu = 3$ and we have

$$T_1 = [I_2 \quad L], \quad D = [\,1 \quad 2\,]$$

Step 2. A solution of (20) with $K_{11}=[1\ \ 2]$ and $K_{12}=[-1\ \ 0]$ is

$$L = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$$

Step 3. The most general solution of T_2 with

$$U = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} -1 & 1 & 1 \\ -1 & 0 & -1 \end{bmatrix}$$

can be determined from (30) as $T'_2 = R + SQ$ where

$$R = 1/6 \begin{bmatrix} 3 & -2 \\ 0 & 2 \\ 3 & 2 \end{bmatrix}, \quad S = 1/6 \begin{bmatrix} 1 & 2 & -1 \\ 2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Step 4. Set up (38) as

$$\begin{bmatrix} 1/2 & -1/3 \\ 1/2 & -1/3 \end{bmatrix} + \begin{bmatrix} 1/6 & 1/3 & -1/6 \\ 1/2 & 1/3 & -1/6 \end{bmatrix} Q = F'$$

Step 5. A possible solution is obtained by a standard pole placement technique as

$$Q = \begin{bmatrix} 3 & -6 \\ 0 & 0 \\ 12 & -8 \end{bmatrix}, \quad F = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}$$

Step 6. The remaining parameters are evaluated from (37), (9), (13) and (15) as

$$T_2 = \begin{bmatrix} -1 & -3 & 3 \\ 0 & 1 & -2 \end{bmatrix}, \quad H = \begin{bmatrix} 3 \\ -2 \end{bmatrix}, \quad G = \begin{bmatrix} -4 & -3 & 6 \\ 1 & 2 & -6 \end{bmatrix}$$

and $E = [2 \quad 3 \quad 2]$.

Remark. For this example an observer of first order can also be constructed. We left this as an exercise for our readers.

VI. CONCLUSION

We have presented a simple design technique to construct stable functional observer of minimal order.

The approach yields a systematic procedure in which the dimension is increased step by step until all observer requirements are satisfied.

Although the structural properties of the pair $(A,[K'C']')$ plays an important role in the design algorithm, it is also interesting to note that the specification of the matrix T enables one to formulate conditions which ensure the arbitrary assignability of the eigenvalues of the minimal order functional observer.

We believe that almost all examples so far published in the literature are of feasible type and can be solved by the present approach more efficiently.

One algorithm is are also included for the design of single functional observer of order $\nu-1$.

REFERENCES

[1] D.G. Luenberger, "Observing the State of a Linear System," IEEE Trans. Mil. Electron, Vol. MIL-8, pp. 74-80, Apr. 1964.

[2] _____, " Observers for Multivariable Systems," IEEE Trans. Automat. Contr., Vol. AC-11, pp. 190-197, Apr. 1966.

[3] _____, "An Introduction to Observers," IEEE Trans. Automat. Contr., Vol. AC-16, pp. 596-602, Dec. 1971.

[4] T.E. Fortmann and D. Williamnson, "Design of Low-order Observers for Linear Feedback Control Laws," IEEE Trans. Automat. Contr., Vol. AC-17, pp. 301-308, June 1972.

[5] P. Murdoch, "Observer Design for a Linear Functional of the State Vector." IEEE Trans. Automat. Contr., Vol. AC-18, pp. 308-310, June 1973.

[6] _____, "Design of Degenerate Observers," IEEE Trans. Automat. Contr., Vol. AC-19, pp. 441-442, Aug. 1974.

[7] J.B. Moore and G.F. Ledwich, "Minimal Order Observers for Estimating Functions of a State Vector," IEEE Trans. Automat. Contr., Vol. AC-20, Oct. 1975.

[8] J.R. Roman and T.E. Bullock, "Design of Minimal Order Stable Observers for Linear Functions of the State via Realization Theory," IEEE Trans. Automat. Contr., Vol. AC-20, Oct. 1975.

[9] S.H. Wang and E.J. Davison, "A Minimization Algorithm for the Design of Linear Multivariable Systems," IEEE Trans. Automat. Contr., Vol. AC-18, June 1973.

[10] W.A. Wolovich, "On the Stability of Solutions to Minimal and Non-minimal Design Problems," IEEE Trans. Automat. Contr., pp. 88, Feb. 1977.

[11] H. Kimura, "Design of Observers for Linear Functions of the State - A Geometric Approach," Department of Computing and Control, Imperial College London, England, Oct. 1975.

[12] _____, "Geometric Structure of Observers for Linear Feedback Control Laws, " IEEE Trans. Automat. Contr., Vol. AC-22, No. 5, Oct. 1977.

[13] K. Furuta and S. Kawaji, "Linear Function Observer with Possibly Minimal Dimension," IEEE Trans. Automat. Contr., Vol. AC-22, pp. 977-980, 1977.

[14] S. Kawaji, "Design Procedure of Observer for the Linear Functions of the State," Int. J. Control, Vol. 32, No. 3, pp. 381-394, 1980.

[15] W.M. Wonham, "Dynamic Observers - A Geometric Theory," IEEE Trans. Automat. Contr., Vol. AC-15, pp. 258-259, June 1970.

[16] _____, "Linear Multivariable Control, A Geometric Approach," Springer Verlag, 1979.

[17] H.R. Sirisena, "Minimal-order Observers for Linear Functions of a State Vector," Int. J. Control, Vol. 29, No. 2, pp. 235-254, 1979.

[18] H. Kajiwara and K. Furuta, "Canonical Realization of Observers with Arbitrarily Assigned Poles for Linear Functions of the State," Int. J. Control, Vol. 29, No. 3, pp. 457-469, 1979.

[19] F.W. Fairman and R.D. Gupta," Design of Multifunctional Reduced Order Observers," Int. J. Systems Sci., Vol. 11, No. 9, pp. 1083-1094, 1980.

[20] B. Gopinath, "On the Control of Linear Multiple Input-Output Systems," Bell Sys. Tech. J., Vol. 50, pp. 1063-1081, 1971.

[21] R.D. Gupta, F.W. Fairman and T. Hinamoto, "A Direct Procedure for the Design of Single Functional Observers," IEEE Trans. Circuits Syst., Vol. CAS-28, No. 4, Apr. 1980.

A Parallel Algorithm for All Pairs Shortest Paths in a Random Graph

Alan Frieze
GSIA, Carnegie-Mellon University and
Department of Computer Science,
Queen Mary College, London

Larry Rudolph
Department of Computer Science,
Carnegie-Mellon University
Pittsburgh, PA 15213

Graph theory has proved to be a useful tool and there is much need for fast algorithms to solve graph problems. We consider here the problem of finding the shortest distance between all pairs of vertices of a random graph whose edge lengths are independent identically distributed nonnegative random variables. Each edge length has a distribution function F with $F(0) = 0$ and that F is differentiable at zero.

We show that with this model of randomness all shortest distances can be computed in $\mathbf{O}(\log\log n)$ expected parallel time using n^3 parallel processors. This compares with $\mathbf{O}(\log^2 n)$ worst-case time, with n^3 processors ([Dekel]). We can, in fact, assume that the edge distances are uniformly distributed in [0,1]. The extension to the general case follows as in Frieze and Grimmett ([Frieze]). Reif and Spirakis ([Reif]) also consider the problem of computing the all pairs shortest paths of a random graph, however, they require that the edge weights satisfy a *triangle inequality* and thus their results do not apply to the general case. Moreover, we require only n^3 processors whereas they require at least n^4 processors.

In our model of computation, we assume that comparisons, arithmetic operations and the generation of random integers all take constant time. In addition, we assume that all the processors share common memory and can read or write to any cell in a single cycle. More precisely, we assume a concurrent-read, concurrent-write ($CRCW$) model of parallel computation; if multiple processors may attempt to store into the same cell at the same time an arbitrary one succeeds

Of notable interest is the part of our algorithm that chooses the minimum of n values quickly. Parallel $CRCW$ algorithms for choosing the minimum with n processors in $\mathbf{O}(\log\log n)$ time ([Valiant]), with $1 < p \leq n$ processors in $\mathbf{O}(n/p + \log\log p)$ time ([Shiloach]), and with n^2 processors in constant time have been well known. Probabilistic decision tree algorithms using only n processors and constant time are also known

Matrix Product Algorithm (MPA)

Input: A matrix D, where $d_{i,j}$ is the length of edge (i,j).
Output: Matrix D^*, where $d^*_{i,j}$ is the shortest distance from i to j.

Begin
  $D^{(1)} := D$
  Repeat
(1) $D^{(k+1)} := D^{(k)} * D^{(k)}$
  Until $D^{(k+1)} = D^{(k)}$
  $D^* := D^k$
End

Figure 1: All Pairs Shortest Path Algorithm

([Reischuk] and [Meggido]). We present a *CRCW* parallel algorithm, i.e. we account for the processor allocation time, for this problem using n processors and show that with high probability, only three constant time steps are needed.

1. The Matrix Product Algorithm

Let D be the original matrix for a graph $G_n = (V,E)$, where d_{ij} is the length of edge (i,j) or ∞ if there is no such edge and let $N = \{1,2,\ldots,n\}, n = |V|$. It is well known that the following 'matrix' algorithm (Figure 1) computes the all pairs shortest distances.

The * operation in line (1) denotes a special type of matrix operation defined as follows:

$$d^{(k+1)}_{ij} = \min\{\{d^{(k)}_{ip} + d^{(k)}_{pj} | p \in N\} \cup D_{ij}\} \quad i,j \in N$$

The validity of MPA follows from the easily demonstrated

$d^{(k)}_{ij}$ is the minimum length of a path from i to j using 2^k edges or less.

Let

$$m(D) = \min\{\ p \mid \text{ for all } i,j \in N \text{ there exists a shortest path from } i \text{ to } j \text{ using } p \text{ edges or less}\}. \tag{2}$$

Clearly line (1) of MPA is executed at most $\lceil \log_2 m(D) \rceil$ times. The following two Lemmas derive a bound on $m(D)$ and show how to execute line (1) in constant parallel time with high probability. To avoid confusion, we shall say that every edge has a weight (instead of saying length), a path has length x to mean that there are x edges in the path, and that a shortest path is one that has the smallest total weight, i.e. the sum of the weights on each edge of the path.

Lemma 1. $\Pr\{m(D) \geq 37 \log^2 n\} = o(n^{-.1})$

Proof We use the following result from [Frieze] which says that if the edge weights are uniformly distributed in the range $(0,1)$ then, with high probability, there are no very heavy shortest paths between nodes:

$$\Pr(\quad \text{there exists } i,j \in N \text{ such that } d^*_{ij} > 12\log n/n) = o(n^{-1}) \tag{3}$$

Let $X = \{(i,j) \in N \times N | i \neq j \text{ and } d_{ij} \leq 1/(3n)\}$ and consider the graph $H = (V,X)$. Note that this is a random digraph where independently each $(i,j) \in N \times N$ is an edge of H with probability $p = 1/(3n)$ and not an edge of H with probability $1-p$.

We next show that, with high probability, H does not contain paths with many edges.

$$\Pr(H \text{ contains a path with } w = \lceil \log_e n \rceil \text{ edges }) = o(n^{-.1}). \tag{4}$$

Indeed the expected number of paths in H with w edges is

$$\leq \binom{n}{w+1}(w+1)!(\frac{1}{3n})^w \leq n(\frac{1}{3})^w = o(n^{-.1}) \tag{5}$$

and (4) follows.

Suppose now that neither of the events described in (3) or (4) occurs. It is left to show that $m(D) \leq 37\log^2 n$, i.e. any path with more than $37\log^2 n$ edges is not a shortest path. Let P be any path in G_n with $t > 37\log^2 n$ edges. By (4) there are no paths in H longer than w edges and so P contains at least $t/(w+1)$ edges not in X. By construction of H, each edge of P not in X has weight greater than $1/(3n)$ and so when n is large,

$$\text{weight of } P \text{ in } G_n \ > (\frac{t}{w+1})(\frac{1}{3n}) > (\frac{12\log n}{n}),$$

and hence P is not a shortest path. ∎

Lemma 2. Line (1) in MPA can be executed in constant expected parallel time using n^3 processors.

Proof. As there are n^3 processors, n processors can be allocated to the calculation of $d^{(k+1)}_{ij}$ for each $i,j \in N$ in parallel. We show there are enough processors so that line (1) of MPA can be executed in constant expected parallel time. Line (1) consists of two steps; first n^2 sets of new shortest distances are computed and second the minimum value in each of these sets is chosen. Note that each set contains at most n elements.

Let us fix i,j,k and write $u_p = d^{(k)}_{ip} + d^{(k)}_{pj}$. Clearly $u_1, u_2, \ldots, u_n$ can be computed in parallel in constant time with n processors for each p; processor PE_p computes u_p for $p = 1,2,\ldots,n$. All that needs to be shown is that the minimum of n numbers can be computed in expected constant time with n processors. This is proved in the following section. ∎

2. Computing the Minimum in Constant Time

In this section we present a parallel algorithm for computing the minimum of n numbers, which we refer to as elements to avoid confusion, in constant expected time. Note that we present an algorithm and not just a decision tree method.

We first review the well known method to compute the minimum of m elements with m^2 processor in constant time. Let $x_1, x_2, \ldots, x_m$ be the m elements. The m^2 processors are used to perform all pairwise comparisons in parallel. A vector of size m is used to record all the x_i for which there is at least one element smaller than it. There will be one x_i for which the corresponding array element has not been set. The algorithm in the figure 2 makes this precise.

Select Minimum

Input: Set of $x_1, x_2, \ldots, x_m$ elements
Output: *minval*

Begin

/* Initialize Data Structures */
$(0 \leq i < m)$ $R[i] \leftarrow$ "Nobody is smaller"

/* Perform All Pairwise Comparisons */
$(0 \leq i, j < m)$ **if** $x_i < x_j$ **then**
$R[i] \leftarrow$ "Somebody is Bigger"

/* Find element that is smallest */
$(0 \leq i < m)$ **if** $R[i] =$ "Nobody is smaller" **then**
minval $\leftarrow x_i$

End

Figure 2: Minimum of m elements with m^2 processors

The improvement in which the minimum of n elements can be found in constant time with n processors is based on a lovely idea independently discovered by Reischuk and Meggido. The idea is to randomly choose square root of n of the elements and compute their minimum in constant time with the n processors according to the above description. With very high probability, there should remain only about $\sqrt{n}$ elements that are smaller than this minimum. A second and possible a third iteration will almost surely find the real minimum of the whole set. Reischuk examined the more general case of choosing the k^{th} largest and presents an upper bound on the expected number of phases of comparison steps (i.e. a decision, tree model). Meggido also assumes a decision tree model and shows that the minimum can be found in constant time "almost surely".

We are concerned with a parallel algorithm and not a decision tree result, and so, we must consider the processor allocation problem. Moreover, we precisely compute exactly the expected number of iterations required to compute the minimum.

Let $m = \lfloor \sqrt{n} \rfloor$ with the n elements denoted $u_1, u_2, \ldots, u_n$. The first step of the decision tree method poses no problem for processor allocation; the first m processors randomly choose elements with the element chosen by PE_i assigned to x_i. After the first iteration, however, there may be many candidate elements remaining although in no apparent order. It is not clear how to assign at most m of the remaining elements to the array x. We propose the following method (see Figure 3):

Instead of choosing a candidate element, u_j to assign to x_i, we invert the procedure and have each candidate element u_j choose a x_i. That is, for each element, u_j, that is smaller than the current minimum processor j randomly chooses an index i and assigns u_j to x_i. If more than one element chooses the same index then we assume that an arbitrary one of the assignments succeed. Since about $1/e$ of the indices will be nonempty, the iteration can proceed choosing a new minimum.

More precisely, we present a parallel program outline, called *APSPA* and then derive its expected running time.

We now proceed to prove that with high probability (*i.e.*$1 - \mathbf{o}(n^{-1})$) only a small number (three or four) of iterations are required to find the minimum. (Note that due to our method of assigning elements to the array Test we cannot use the proofs of Meggido or Reischuk.)

Let T_k denote the set of elements successfully stored in Test during the kth iteration and let C_k denote the set of candidates, that is $C_k = \{u_i < minval_{k-1}\}$, where $minval_{k-1}$ is the value of *minval* after the $k-1$ iteration. We examine the probable sizes of these sets and prove:

Theorem 1: The following five statements about the sets Test and Candidates are all true. (We precede each statement with an informal, intuitive description in parentheses.)

(a) (With high probability, during the first iteration, all of Test gets filled.)

$$\Pi_A = \Pr(|T_1| < m) = \mathbf{o}(n^{-1})$$

(b) (With high probability, the minimum chosen at the end of the first iteration is no larger than $2m\log n$ of the elements.)

$$\Pi_B = \Pr(|C_2| \geq 2m\log n \text{ given that } |T_1| = m) = \mathbf{o}(n^{-1})$$

(c) (With high probability, during the second iteration, 1/4 of Test gets filled or, if C_1 is small then Test gets at least 1/4 of C_1.)

$$\Pi_C = Pr(|T_2| \leq .25\mathbf{min}(m, |C_2|) \text{ given that } |C_2| < 2m\log n) = \mathbf{o}(n^{-1})$$

(d) (With high probability, the minimum chosen at the end of the second iteration is no larger than $17\log^2 n$ of the elements.)

$$\Pi_D = \Pr(\ |C_3| \geq 17\log^2 n \text{ given that } |C_2| < 2m\log n \text{ and } |T_2| > .25\mathbf{min}(m, |C_2|)) = \mathbf{o}(n^{-1})$$

(e) (With high probability, during the third iteration all but at most 3 of the remaining candidates are assigned to Test.)

$$\Pi_E = \Pr(|C_3 - T_3| \geq 3 \text{ given that } |C_3| < 17\log^2 n) = \mathbf{o}(n^{-1})$$

We deduce then that with high probability after three iterations the set of candidates is reduced to at most two elements and thus the minimum will be found after no more than four iterations.

Proof. of (a): Consider during the first iteration how elements are assigned to the array Test. The probability that some index of Test is not chosen is:

$$\Pr(\text{ Test[i] = nil }) = (1 - 1/m)^n \leq e^{-m}$$

and so

$$\begin{aligned} \Pi_A &= \Pr(|T_1| < m) \\ &\leq \textstyle\sum_{i=1}^{m} \Pr(\text{ Test[i] = nil }) \leq me^{-m} \end{aligned}$$

∎

Improved Minimum Selection

Input: n numbers $u_1, u_2, \ldots, u_n$ and n PE's,
Output: the minimum value, *minval*.

Loop
/* Initialize Data Structures */
$(0 < i \leq m)$ Test[i] := **nil**
IsMin[i] := "Nobody is smaller"

$(i = 1)$ Done := **true**
minval := ∞

/* Choose m Elements */
$(0 < i \leq n)$ **if** $u_i <$ *minval* **then**
r_i := Random Index between 1 and m
Test[r_i] := u_i

/* Compare all pairs and record results */
$(0 < ij \leq m)$ **if** Test[i] < Test[j] **then**
IsMin[j] := "Somebody is smaller"

/* Pick minimum */
$(0 < i \leq m)$ **if** IsMin[i] = "Nobody is smaller" **then**
minval := Test[i].

/* Are we done? */
$(0 < i \leq n)$ **if** $u_i <$ *minval* **then** done := **false**
Until Done

Figure 3: Minimum of n elements with n processors

Proof. of (b): This result follows from the fact that the number of ways to form the array Test without choosing any of the $2m\log n$ smallest elements is very small compared to the number of ways to form it choosing from all n elements. More formally:

$$T_1 \text{ is a random } m\text{-subset of } N \text{ and if } \quad |C_2| \geq s = \lceil 2m\log n \rceil$$

then there are only $\binom{n-s}{m}$ choices for T_1. Thus

$$\Pi_B \leq \binom{n-s}{m} / \binom{n}{m} \leq (1 - s/n)^m \leq e^{-sm/n} < e^{-2\log n} = o(n^{-1})$$

∎

Proof. of (c): If $|\ C\ |$ is big then we expect about 1/e of Test to be filled. We use the value 1/4 since it is less than 1/e and also works for when C is small. Also note that the summation is begun from 4 since the probabilities for $k < 4$ are all zero.

$$\Pi_C \leq \sum_{k=4}^{s-1} \Pi_k / \Pr(|C_2| < s)$$

where

$$\Pi_k = \Pr(|T_2| \leq .25\min(m,k) \text{ given that } |C_2| = k)$$

Let $k_1 = \lfloor k/4 \rfloor$, then for $k \leq m$,

$$\begin{aligned}
\Pi_k &\leq \Pr(\text{ there exists a } k_1\text{-subset } S \text{ of } \{1,2,\ldots,m\} \\
&\quad \text{such that for each } i \in C_2 \text{ processor } PE_i \text{ chooses a random integer in } S) \\
&\leq \binom{m}{k_1}(k_1/m)^k \\
&\leq (m\,e/k_1)^{k_1}(k_1/m)^k \\
&\leq (k\,e^{1/3}/4m)^{3k/4}.
\end{aligned}$$

Also $k > m$ implies $\Pi_k \leq \Pi_m$ and so

$$\Pi_C \leq \left(\frac{1}{\Pr(|C_2| < s)}\right) \sum_{k=4}^{minval} (k\,e^{1/3}/4m)^{3k/4} + 2m\log n(e^{1/3}/4)^{3m/4} = o(m^{-3}).$$

∎

Proof. of (d): We will use the following shorthand notation:

$$\begin{aligned}
p &: \lceil 17\log^2 n \rceil \\
A_k &: (|C_2| = k) \\
B_j &: (|T_2| = l) \\
W_c &: (|C_2| < 2m\log n) \\
W_t &: (|T_2| > \min\{m, |C_2|\}/4)
\end{aligned}$$

Using this notation, we rewrite and proceed as follows:

$$\begin{aligned}
\Pi_D &= \Pr(|C_3| \geq p \mid W_c \;\&\; W_t) \\
&= \textstyle\sum_{k=p}^{2m\log n} \sum_{l=.25\min\{m,|k|\}}^{k} \frac{\Pr(|C_3|\geq p \mid A_k \;\&\; B_l)\,\Pr(A_k \;\&\; B_l)}{\Pr(W_c \;\&\; W_t)} \\
&\leq \textstyle\sum_{k=p}^{2m\log n} \sum_{l=.25\min\{m,|k|\}}^{k} \binom{k-p}{l}/\binom{k}{l} \\
&\leq \textstyle\sum_{k=p}^{2m\log n} \sum_{l=.25\min\{m,|k|\}}^{k} (1-p/k)^l \leq \sum\sum e^{-17\log n/8}
\end{aligned}$$

and (d) follows. ∎

Proof. of (e): Imagine that $r_i, i \in C_3$, elements are chosen sequentially. Then $|C_3 - T_3|$ is the number of times an r_i, just generated, has been chosen before. But the probability of this happening is never more than p/m regardless of the previous values of the r_i's. As $C_3 < p$ we have

$$\Pr(|(C_3 - T_3)| \geq 3) \leq \sum_{t=3}^{p} \left(\binom{p}{t}\right)(p/m)^t(1-p/m)^{p-t} = O(p^6/m^3)$$

(the terms in the sum reduce by at least a factor of p^2/m as k increases.) ∎

The result of this paper can be summed up as follows:

Theorem 2: The expected running time of MPA is $O(\log\log n)$.

Proof. From Lemmas 1 and 2, the expected running time is $O(\log(37\log^2 n) + n^{-.1}\log^2 n)$. ∎

Bibliography

Dekel, E., D. Nassimi, S. Sahni, "Parallel Matrix and Graph Algorithms," *SIAM Journal of Computing*, Vol. 10, (1981).

Frieze, A. and G. R. Grimmett, "The Shortest Path Problem for Graphs with Random Arc-Lengths" *to appear in Discrete Applied Mathematics.*

Megiddo, N. "Parallel Algorithms for Finding the Maximum and the Median Almost Surely in Constant Time," Preliminary Manuscript, GSIA, Carnegie-Mellon University, Pittsburgh, PA (1982).

Reif, J. and P. Spirakis, "Expected Parallel Time and Sequential Space Complexity of Graph and DiGraph Problems," Technical Report No. 91, New York University, Oct. 1983.

Reischuk, R., "A Fast Probabilistic Parallel Sorting Algorithm," *22nd Symposium on Foundations of Computer Science*, (1981), 212-219.

Shilaach, Y. and V. Vishkin, "Finding the Maximum, Merging, and Sorting in a Parallel Computation Model," *Journal of Algorithms*, Vol. 2, (1981), 88-102.

Valiant, L. G., "Parallelism in Comparison Problems," *SIAM Journal of Computing*, Vol. 4, No. 3, (1975) 348-355.

PARALLEL APPROXIMATION SCHEMES FOR SUBSET SUM AND KNAPSACK PROBLEMS

JOSEPH PETERS
Department of Computing Science
Simon Fraser University

LARRY RUDOLPH
Department of Computer Science
Carnegie-Mellon University

ABSTRACT

This paper presents parallel approximation schemes for the *Subset Sum*, *0-1 Knapsack*, and several other optimization problems. These algorithms offer a three-way trade-off among parallel time, the accuracy of the solution, and the number of processors used. The maximum numbers of processors which can be usefully employed depend on n (the size of the input), and the accuracy requirement ϵ. The parallel running times of the algorithms are polynomial in both $\log n$ and $\log(\frac{1}{\epsilon})$ when enough processors are used.

1. INTRODUCTION

The *0-1 Knapsack* and *Subset Sum* problems are members of a large class of discrete optimization problems which have been shown to be *NP*-hard (cf. [2]). The widely held belief that no *NP*-complete problem can be solved in polynomial time has generated considerable interest in the study of "fast" approximation algorithms which find "good", but not necessarily optimal, solutions to instances of *NP*-hard optimization problems. For some *NP*-hard optimization problems, including 0-1 Knapsack and Subset Sum, algorithms called *approximation schemes* have been developed ([3], [4], [5], [6], [7], [8], [9], [11], [13], [14]). An approximation scheme is an algorithm which inputs an instance of a problem and an accuracy requirement ϵ, and returns an approximate solution with relative error at most ϵ. The running time is a function of both the size of the instance and $\frac{1}{\epsilon}$. Thus, an approximation scheme offers a trade-off between time and accuracy.

Approximation algorithms reduce the time required to solve "hard" optimization problems by relaxing the requirement that the solution must be optimal. A second way to decrease the time to find a usable solution is to distribute the work among several parallel processors. This paper combines these two approaches and presents parallel approximation schemes for several optimization problems. These algorithms offer a three-way trade-off among parallel time, the accuracy of the solution, and the number of processors used. They are the first algorithms to exhibit such trade-offs. The maximum numbers of processors which can be usefully employed depend on n (the size of the input), and the accuracy requirement ϵ. The parallel running times of the algorithms are polynomial in both $\log n$ and $\log(\frac{1}{\epsilon})$ when enough processors are used.

The algorithms in this paper are designed for shared-memory, parallel models of computation such as the PRAM of Fortune and Wyllie [1]. The model has p identical and independent processing elements (PE's) and a global memory to which all PE's have access. Concurrent reads are allowed and are assumed to take constant time. Concurrent writes to the same memory location are not allowed, but two or more processors can write concurrently to different memory locations (in constant time). The algorithms can be executed either synchronously or asynchronously. If asynchronous execution is chosen, then it is assumed that some synchronization method is available and that all processors can be synchronized in constant time.

The next section of this paper contains background material and definitions. In Section 3, a parallel approximation scheme for the Subset Sum problem is presented. The algorithm runs in parallel time $O(\frac{n}{p\epsilon}\log(\frac{1}{\epsilon}))$ using $1 \leq p \leq \frac{8}{\epsilon}$ PE's for any $0 < \epsilon \leq 1$. The time is then improved to $O(\log n \log(\frac{1}{\epsilon}))$ using $O(\frac{n}{\epsilon^2})$ PE's. Section 4 contains an $O(\frac{n^2}{p\epsilon})$ approximation scheme for 0-1 Knapsack which uses $1 \leq p \leq \frac{2n}{\epsilon}$ PE's. The algorithm is then improved to run in $O(\log^2 n + \log n \log(\frac{1}{\epsilon}))$ time using $O(\frac{n^2}{\epsilon^2})$ PE's, and is generalized to an $O(\log^2(\frac{n}{\epsilon}))$ approximation scheme for the *Unbounded Knapsack* problem. The techniques of this paper can be used to develop parallel approximation schemes for many other problems.

Parts of this research were done while both authors were at the Department of Computer Science, University of Toronto.

2. PRELIMINARIES

Definition 1. The *0-1 Knapsack problem* is: given a finite set $E_n = \{1,2,\ldots,n\}$ of objects, a *weight* $a_i \in \mathbf{Z}^+$ and a *profit value* $p_i \in \mathbf{Z}^+$ for each $i \in E_n$, and a *knapsack capacity* $b \in \mathbf{Z}^+$, find a subset $S \subseteq E_n$ that maximizes the *objective function* $\sum_{i \in S} p_i$ subject to the *constraint* $\sum_{i \in S} a_i \leqslant b$.

Informally, the 0-1 Knapsack problem is the problem of maximizing the profit value of the contents of a container of finite capacity. It can be generalized and restricted in a variety of ways. An interesting special case of 0-1 Knapsack is the *Subset Sum* problem in which the profit value of each object is the same as its weight. When $b = \frac{1}{2}\sum_{i=1}^{n} a_i$, the Subset Sum problem is equivalent to the problem of distributing a set of n tasks with known processing times to two identical processors in such a way that the elapsed time to process all tasks is minimized. The *Unbounded Knapsack* problem is a generalization of 0-1 Knapsack which allows multiple copies of objects to be placed in the knapsack.

A *feasible solution* for an instance of an optimization problem is any subset which satisfies the constraint. Without loss of generality, it can be assumed that single objects are feasible (i.e., $a_i \leqslant b$; $1 \leqslant i \leqslant n$). An *optimal solution* for a *maximization problem* is a feasible solution which maximizes the value of the objective function. All other feasible solutions are *approximate solutions*. The most common way to measure the accuracy of approximate solutions is with the *relative error* measure.

Definition 2. Let I be an instance of a maximization problem with objective function f, optimal solution S_0, and let S be any feasible solution for I. The *relative error* of S as a solution for I is $r(S) = \frac{f(S_0) - f(S)}{f(S_0)}$. It is assumed that $f(S_0) > 0$. For maximization problems, $0 \leqslant r(S) \leqslant 1$ for every feasible solution S, and the relative error of optimal solutions is 0.

Definition 3. An algorithm which, for every instance of a problem, returns a feasible solution, S, with $r(S) \leqslant \epsilon$ is called an *ϵ-approximate algorithm*. An algorithm A is an *approximation scheme* if it takes as input a pair (I,ϵ) where I is an instance and $0 < \epsilon \leqslant 1$ is a constant, and for every pair (I,ϵ) it returns a feasible solution S with $r(S) \leqslant \epsilon$. If A runs in time polynomial in both the size of I and $\frac{1}{\epsilon}$, then A is a *fully polynomial approximation scheme* (FPAS).

Definition 4. $f(n,\frac{1}{\epsilon}) = O(g(n,\frac{1}{\epsilon}))$ if there are constants $c_1,c_2 > 0$ such that for all $n \geqslant c_1$ and all $0 < \epsilon \leqslant 1$, $f(n,\frac{1}{\epsilon}) \leqslant c_2 g(n,\frac{1}{\epsilon})$.

To be consistent with the literature on approximation schemes, the uniform cost criterion will be used to analyze algorithms in this paper. Thus, it is assumed that profit values, weights, capacities, and the value of an optimal solution are small enough to fit in a computer word and arithmetic operations on them take constant time. In practice, this is usually the case because more precision is usually not realistic. It is also assumed that $\frac{1}{\epsilon} \leqslant 2^n$ so that an encoding of an instance with n objects has length $O(n)$. Time-accuracy trade-offs for the problems discussed in this paper are not too interesting when $\frac{1}{\epsilon} > 2^n$ because they all can be solved optimally in time $O(2^n)$ or $O(n2^n)$ by enumeration.

3. THE SUBSET SUM PROBLEM

Algorithm 1 below is a FPAS for the Subset Sum problem by Gens and Levner [3]. The algorithm is $O(\frac{n}{\epsilon})$ and is the basis for the parallel approximation schemes in this section. $O(\frac{n}{\epsilon})$ is currently the best time-accuracy trade-off for the Subset Sum problem. Algorithm 1 starts by computing a bound B on $f(S_0)$ such that $B \leqslant f(S_0) \leqslant 2B$. Lawler [9] has shown how to use a fast median finding algorithm to compute B in time $O(n)$. Each *stage* of Algorithm 1 consists of an *expand* phase, and a *compact* phase. In the k^{th} expand phase, a_k is added to each partial solution from stage $k-1$ to get a new set of partial solutions. The new partial solutions are then merged with the set of partial solutions from stage $k-1$. In the compact phase, some partial solutions are discarded in a way which ensures that the values of no three partial solutions are within ϵB of each other, so, there are never more than $O(\frac{1}{\epsilon})$ partial solutions to consider.

Three arrays, each of size at most $O(\frac{1}{\epsilon})$, are used by Algorithm 1. All three arrays are maintained in non-decreasing order of partial solution value. The arrays are subscripted to emphasize the step during which their values were generated. (The subscripts are used in the proofs; they do not appear in the algorithms.) The arrays are:

T_k: Contains all partial solutions that may lead to a final solution. It contains at most $\frac{4}{\epsilon}$ elements.

U_k: Contains the set of partial solutions from expand phase k (i.e., the result of adding element a_k to each element of T_{k-1}).

V_k: T_{k-1} merged with U_k. It may contain up to $\frac{8}{\epsilon}$ elements.

The notation $T[i]$ (similarly $U[i]$ and $V[i]$) means the i^{th} element of array T, and $T_k[i]$ means the i^{th} element of T at the end of stage k. The array U in the expand phase is not really necessary; the steps in the loop which generates U, and the steps of the MERGE can be interleaved to produce array V directly. (This is done in the parallel version below.) Array U is used to simplify the presentation.

After the algorithm terminates, the largest element in T is returned as the value of an ϵ-approximate solution. A trace-back routine can be used to determine which elements a_i are in the solution [9]. The auxiliary data structures used in the trace-back routine in [9] can be used in all of the parallel algorithms in this paper without affecting the time bounds obtained. Furthermore, the elements in the solution can be identified in $O(\log n)$ time using n PE's and the marking technique described in Lemma 5 later in this section. Since this marking will not increase any of the time bounds either, the remainder of this paper will concentrate on finding the values of ϵ-approximate solutions.

Algorithm 1. Fully Polynomial Approximation Scheme for Subset Sum

Compute B such that $B \leq f(S_0) \leq 2B$

T[1] ← 0

FOR $k \leftarrow 1,2,\ldots,n$

```
    /* Expand Phase */
    j ← 1
    WHILE j ≤ |T| and T[j] + a_k ≤ b
        U[j] ← T[j] + a_k
        j ← j + 1
    V ← MERGE(T,U)

    /* Compact Phase */
    i ← 1          j ← 2
    WHILE j ≤ |V|
        IF V[j] = T[i] THEN j ← j + 1    /* Eliminate duplicates */
        ELSE
            i ← i + 1          T[i] ← V[j]          j ← j + 1
            WHILE j ≤ |V| and T[i − 1] + εB ≥ V[j]
                T[i] ← V[j]
                j ← j + 1
```

Informally, the compact phase of Algorithm 1 moves the following set of partial solutions into T and eliminates the rest: $T[1] = V[1] = 0$, $T[2] = V[i]$ where $V[i]$ is the largest element of V such that $T[1] + \epsilon B \geq V[i] > T[1]$ (if there is no such $V[i]$, then $T[2] = V[i']$ where $V[i']$ is the smallest element such that $V[i'] > T[1]$), $T[3] = V[j]$ where $V[j]$ is the largest element of V such that $T[2] + \epsilon B \geq V[j] > T[2]$ (if there is no such $V[j]$, then $T[3] = V[j']$ where $V[j']$ is the smallest element such that $V[j'] > T[2]$), and so on. The last element of T is the element of V with largest value. Note that the expand phase will always try to put the partial solution with value a_k into U_k because $T[1] = V[1] = 0$ is always retained. Also note that there can never be three partial solutions with the same value in array V because duplicates are eliminated at the end of every stage. It is not hard to show that each stage is $O(\frac{1}{\epsilon})$ and that the entire algorithm is

$O(\frac{n}{\epsilon})$ [3]. A proof that Algorithm 1 finds an ϵ-approximate solution appears in [12].

Algorithm 2 is a parallel version of Algorithm 1 which uses $p = \frac{8}{\epsilon}$ PE's. It will be proved later that Algorithm 2 retains the same set of partial solutions as Algorithm 1, so, at any time there are at most $\frac{8}{\epsilon}$ partial solutions to be considered. The case $1 \leqslant p < \frac{8}{\epsilon}$ is considered later in this section. During stage k, each PE_i expands the i^{th} partial solution by adding item a_k to $T_{k-1}[i]$. Using binary searches requiring $O(\log(\frac{1}{\epsilon}))$ time, it then places the new partial solution, $T_{k-1}[i] + a_k$, and the old partial solution, $T_{k-1}[i]$, into V_k. The compact phase, which moves a subset of the elements from V_k to T_k in preparation for the next stage uses an array, *Next*, of pointers. The compact phase requires $O(\log(\frac{1}{\epsilon}))$ time and will be explained in detail below. The algorithm can be executed either synchronously or asynchronously. If it is executed asynchronously, then all processors must synchronize at the indicated places. (I.e., no processor can proceed past a *synchronization point* until all processors have reached that point.) It is assumed that each such synchronization requires constant time.

Algorithm 2. Parallel Approximation Scheme for Subset Sum

Compute B such that $B \leqslant f(S_0) \leqslant 2B$

Each PE_i executes the following code (note that the variable $i \geqslant 1$ is the PE identifier). Each PE has two private variables j and j'. SEARCH(X,y) returns an index l into array X such that $X[l] < y \leqslant X[l+1]$. If $X[t]$ is the largest element of X and $X[t] < y$ then t is returned. If $y \leqslant X[1]$ then 0 is returned.

```
T[1] ← 0          T[i] ← undefined for i > 1
FOR k = 1,2,...,n
    <Synchronize>
    /* Expand Phase */
    V[i] ← undefined
    IF T[i] is defined THEN
        j ← SEARCH(T,T[i] + a_k)
        IF T[i] + a_k = T[j+1] THEN j ← j + 1
        IF T[i] + a_k ⩽ b THEN V[i+j] ← T[i] + a_k          (†)
        j' ← SEARCH(T,T[i] − a_k)
        V[i+j'] ← T[i]                                      (††)
    <Synchronize>
    /* Compact Phase */
    Next[i] ← null
    IF V[i] is defined and V[i+1] is defined THEN
        Next[i] ← i + 1
        WHILE V[Next[i] + 1] is defined and V[i] + εB ⩾ V[Next[i] + 1]
            Next[i] ← Next[i] + 1
        IF V[i] = V[Next[i]] THEN Next[i] ← Next[i] + 1   /* Eliminate duplicates */
        IF V[Next[i]] is undefined THEN Next[i] ← null
    T[i] ← undefined
    j ← index of V[i] in array T (see below)                (†††)
    IF j > 0 THEN T[j] ← V[i]
```

Theorem 1. Algorithm 2 has parallel time complexity $O(n\log(\frac{1}{\epsilon}))$, uses $O(\frac{1}{\epsilon})$ PE's, and produces an ϵ-approximate solution for any $0 < \epsilon \leqslant 1$.

Four technical lemmas are needed to prove Theorem 1. The proofs of the first three lemmas are mechanical and appear in [12]. Lemma 2 proves that line (†) places the partial solution $T[i] + a_k$ into the correct location in array V. Lemma 3 serves a similar purpose for line (††). Lemma 4 proves that the computation of array *Next* takes constant time. Lemma 5 shows that line (†††), which is used to move elements of V back into T can be implemented in $O(\log(\frac{1}{\epsilon}))$ time and that Algorithm 2 retains the same set of partial solutions at the end of each stage as Algorithm 1. Note that the partial solutions $T[i] + a_k$; $1 \leq i \leq |T|$, generated during the k^{th} expand phase are merged directly into array V_k. Thus, the array U_k which was used in Algorithm 1 to store the set of new partial solutions at stage k is not used in Algorithm 2.

Lemma 2. During stage k, if j is such that $T_{k-1}[j] \leq T_{k-1}[i] + a_k < T_{k-1}[j+1]$ at line (†), then $V_k[i+j] = T_{k-1}[i] + a_k$.

Lemma 3. During stage k, if j' is such that $T_{k-1}[j'] + a_k < T_{k-1}[i] \leq T_{k-1}[j'+1] + a_k$ at line (††), then $V_k[i+j'] = T_{k-1}[i]$.

Lemma 4. $V_k[i] + \epsilon B < V_k[i+4]$; $1 \leq i \leq |V| - 4$; $1 \leq k \leq n$.

Lemma 5. Line (†††) can be implemented in $O(\log(\frac{1}{\epsilon}))$ parallel time.

Proof. Consider the directed acyclic graph whose nodes correspond to the elements of V_k and whose edges correspond to *Next* (i.e., there is an edge from node $V_k[i]$ to node $V_k[Next[i]]$). Each node in this graph has out-degree one. The node corresponding to $V_k[1]$ is the *root* node. $Next[i]$ points to the next element which should be placed into T_k given that $V_k[i]$ is placed in T_k. It is not hard to see that, if $V_k[1]$ is retained, all nodes reachable from $V_k[1]$ are retained, and all other elements are discarded during the compact phase, then Algorithm 2 will place the same set of partial solution into T_k as Algorithm 1. Furthermore, the distance from the root node (i.e., the number of edges plus 1) is the correct position of the node in T_k. Thus the problem is how to perform fast marking and counting in a graph with out-degree one. This can be done by "marking" $V_k[1]$ and then using the "recursive-doubling" technique of Fortune and Wyllie [1] to mark all nodes reachable from $V_k[1]$. Since $|V_k| = O(\frac{1}{\epsilon})$, this will take $O(\log(\frac{1}{\epsilon}))$ time with $O(\frac{1}{\epsilon})$ PE's. ■

Proof of Theorem 1. Lemmas 2, 3, and 5 establish the correctness of lines (†), (††), and (†††). The remainder of the error analysis follows directly from the analysis of Algorithm 1. Therefore the algorithm produces an ϵ-approximate solution. The bound B is computed in time $O(n)$ with 1 PE using the method in [9]. To establish the time complexity of the remainder of the algorithm, it is sufficient to show that each of the n stages takes $O(\log(\frac{1}{\epsilon}))$ time. As in Algorithm 1, $|T_k| = O(\frac{1}{\epsilon})$ and $|V_k| = O(\frac{1}{\epsilon})$ for $1 \leq k \leq n$. The binary searches in the expand phase therefore take $O(\log(\frac{1}{\epsilon}))$ time. Line (†††) is $O(\log(\frac{1}{\epsilon}))$ by Lemma 5 and the computation of the array *Next* in the compact phase is $O(1)$ by Lemma 4. The remaining steps take constant time. ■

Corollary 6. An ϵ-approximate solution for an instance of the Subset Sum problem can be found in $O(\frac{n}{p\epsilon}\log(\frac{1}{\epsilon}))$ parallel time with $1 \leq p \leq \frac{8}{\epsilon}$ PE's for any $0 < \epsilon \leq 1$.

Proof. If $1 \leq p < \frac{8}{\epsilon}$ then each PE in Algorithm 2 is in charge of more than one index of T and V and each of the n steps then requires $O(\frac{1}{p\epsilon}\log(\frac{1}{\epsilon}))$ time. ■

It should be noted that no write conflicts can occur in Algorithm 2. Lemmas 2 and 3 prove that each of the PE's is writing to a different location in array V in lines (†) and (††) (respectively). Since the method in [1] does not have any write conflicts, Lemma 5 proves a similar statement for the line after line (†††). $Next[i]$ is only updated by PE_i.

Algorithm 2 still contains a large serial component; items are added to the partial solution set one at a time. Algorithm 3 below is a sublinear time approximation scheme for the Subset Sum problem which uses $O(\frac{n}{\epsilon^2})$ PE's. The basic idea of Algorithm 3 is to use divide-and-conquer with enough PE's so that the divided parts can be solved in parallel. First, a bound B is computed: sort the input items $a_1, \ldots, a_n$ into

non-increasing order, compute the n partial sums $\sum_{i=1}^{j} a_i$; $1 \leq j \leq n$, and compare the largest partial sum not exceeding b to $\max_{1 \leq i \leq n} \{a_i\}$. The larger of the two values is B [9]. Now, divide the n items into two disjoint sets of $\frac{n}{2}$ items each, solve the problem for each set recursively, and then combine. The combining will require $\frac{1}{\epsilon^2}$ PE's as explained in Theorem 7 below. At the end of the algorithm, array D corresponds to array T_n in Algorithm 2, and the largest element of D is the value of the approximate solution. A proof that this approximate solution is ϵ-approximate appears in [12].

Algorithm 3. Sublinear Parallel Approximation Scheme for Subset Sum

1. Solve Subset Sum recursively and in parallel on the sets $\{a_1, \ldots, a_{n/2}\}$ and $\{a_{n/2+1}, \ldots, a_n\}$. The result will be two arrays of partial solutions, say A_1 and A_2, each of size at most $\frac{4}{\epsilon}$. (A_1 and A_2 correspond to array T in Algorithm 2.)
2. For each partial solution $A_1[i]$, generate an array $B_i = \{A_1[i] + A_2[j] \mid A_1[i] + A_2[j] \leq b;\ j = 1,2,\ldots\}$.
3. Merge the arrays B_i; $i = 1,2,3,\ldots$ into one array C by sorting. C corresponds to the array V in Algorithm 2 and contains at most $\frac{16}{\epsilon^2}$ elements.
4. Compact C using the method of Algorithm 2 to get an array D containing at most $\frac{4}{\epsilon}$ elements.

Theorem 7. Algorithm 3 has parallel time complexity $O(\log n \log(\frac{1}{\epsilon}))$, uses $O(\frac{n}{\epsilon^2})$ PE's, and produces an ϵ-approximate solution for any $0 < \epsilon \leq 1$.

Proof. The sorting step for computing the bound B takes $O(\log n)$ time with n PE's using the method in [10]. The n partial sums can be computed in $O(\log n)$ parallel time using n PE's, and an $O(\log n)$ binary search using 1 PE will find B. To prove that the parallel running time of the algorithm is $O(\log n \log(\frac{1}{\epsilon}))$ using at most $O(\frac{n}{\epsilon^2})$ PE's, note that there are $\log n$ parallel divide-and-combine steps and at most $\frac{n}{2}$ subproblems are being solved in each step. As the following analysis shows, each subproblem can be solved in $O(\log(\frac{1}{\epsilon}))$ parallel time using $O(\frac{1}{\epsilon^2})$ PE's. Therefore, the entire problem can be solved in $O(\log n \log(\frac{1}{\epsilon}))$ parallel time with $O(\frac{n}{\epsilon^2})$ PE's. To see that each subproblem can be solved in $O(\log(\frac{1}{\epsilon}))$ parallel time, observe that each B_i array in step 2 can be formed in constant time using $\frac{4}{\epsilon}$ PE's, and there are at most $\frac{4}{\epsilon}$ such arrays. Therefore, the array C resulting from step 3 contains at most $\frac{16}{\epsilon^2}$ partial solutions, and the sorting can be done in $O(\log(\frac{1}{\epsilon^2}))$ parallel time with $O(\frac{1}{\epsilon^2})$ PE's using the method in [10]. Step 4 requires $O(\log(\frac{1}{\epsilon^2}))$ parallel time with $O(\frac{1}{\epsilon^2})$ PE's using the method of Algorithm 2. It follows that each subproblem can be solved in $O(\log(\frac{1}{\epsilon^2})) = O(\log(\frac{1}{\epsilon}))$ parallel time using $O(\frac{1}{\epsilon^2})$ PE's. ■

4. THE KNAPSACK PROBLEM

This section presents an $O(\frac{n^2}{p\epsilon})$ parallel approximation scheme for the 0-1 Knapsack problem, where $1 \leq p \leq \frac{2n}{\epsilon}$. The divide-and-conquer technique is then used to obtain a sublinear time algorithm. The parallel algorithms are based on the following serial FPAS which uses the *static interval partitioning* technique of Sahni [13] and the bound computing method of Lawler [9] to obtain an $O(\frac{n^2}{\epsilon})$ FPAS. Lawler's $O(\min\{\frac{n^2}{\epsilon}, \frac{n}{\epsilon^2}\})$ bound ([9]) is currently the best time-accuracy trade-off for 0-1 Knapsack.

In Algorithm 4, partial solutions are stored as pairs (P,A) where P is the sum of the profit values of the objects in the partial solution and A is the corresponding sum of the weights. The *expand* phase of Algorithm 4 is similar to Algorithm 1: U_k is formed from T_{k-1} by adding the object (p_k,a_k) to each partial solution $(P,A) \in T_{k-1}$. Each new partial solution is tested for feasibility during this process and infeasible partial solutions are eliminated. During the *compact* phase, the sets T_{k-1} and U_k are merged to form T_k, and some of the partial solutions are eliminated. The largest profit value in T_n at the end of the algorithm is returned as the value of an approximate solution.

Algorithm 4. Fully Polynomial Approximation Scheme for 0-1 Knapsack

Compute B such that $B \leqslant f(S_0) \leqslant 2B$

Divide the interval [0,2B] into subintervals of size $\frac{\epsilon B}{n}$

Initialize $T_0 \leftarrow \{(0,0)\}$

FOR $k \leftarrow 1,2,\ldots,n$

/* *Expand Phase* */

FOR each $(P,A) \in T_{k-1}$

IF $A + a_k \leqslant b$ **THEN** $U_k = U_k \bigcup \{(P + p_k, A + a_k)\}$

/* *Compact Phase* */

$T_k \leftarrow$ MERGE(T_{k-1},U_k) retaining only the pair with the smallest weight in each subinterval

---

Algorithm 4 first computes a bound B on $f(S_0)$. Since $f(S_0) \leqslant 2B$, the profit values of all feasible solutions must lie in the interval $[0,2B]$. This interval is divided into $\frac{2n}{\epsilon}$ subintervals of size $\frac{\epsilon B}{n}$. During the MERGE step, all partial solutions, except the one with the smallest weight (A value) in each subinterval, are discarded. See [13] for a proof that Algorithm 4 is an $O(\frac{n^2}{\epsilon})$ FPAS.

Algorithm 5 below is a parallel version of Algorithm 4. As in Algorithm 4, a bound B such that $B \leqslant f(S_0) \leqslant 2B$ is computed and the profit space interval $[0,2B]$ is divided into subintervals of size $\frac{\epsilon B}{n}$. Two shared arrays of $\frac{2n}{\epsilon} + 1$ elements are maintained. Array A contains the sums of the a_i's of partial solutions and array P contains the corresponding sums of the p_i's. Array elements $A[i]$ and $P[i]$ are used to store a partial solution with profit value in the subinterval $[\frac{(i-1)\epsilon B}{n}, \frac{i\epsilon B}{n})$; $1 \leqslant i \leqslant \frac{2n}{\epsilon} + 1$. For simplicity, assume for now that there are $\frac{2n}{\epsilon}$ PE's; PE_i is assigned to array elements $A[i]$ and $P[i]$; $1 \leqslant i \leqslant \frac{2n}{\epsilon}$. It is not necessary to assign a PE to elements $A[\frac{2n}{\epsilon} + 1]$ and $P[\frac{2n}{\epsilon} + 1]$ because any partial solution in this subinterval must have a profit value of exactly $2B$ and this would be optimal.

During stage j of the algorithm, each PE checks its subinterval to see if it contains a partial solution. If the subinterval contains a partial solution, then the PE expands it by adding the element (p_j,a_j) to it and computes the index, *New1*, of the subinterval in which the new partial solution belongs. During the compact phase, each PE which has computed a new partial solution attempts to place it into the proper subinterval. If the subinterval is empty, or the weight of the partial solution in the subinterval is larger than the weight of the new partial solution, then the new partial solution is retained; otherwise it is discarded.

The compact phase is split into two parts. First the PE's with odd indices attempt to update arrays P and A and then the PE's with even indices attempt to update P and A. This is done to avoid write conflicts in lines (†) and (††). After the end of each stage, each subinterval contains at most one partial solution. A pair of partial solutions in consecutive subintervals may have profit values less than $\frac{\epsilon B}{n}$ apart. When the same element (p_j,a_j) is added to these partial solutions, they may both belong in the same new subinterval and a write conflict could occur. However, a pair of partial solutions one subinterval apart must have profit values more than $\frac{\epsilon B}{n}$ apart and the new partial solutions must belong in different subintervals.

Algorithm 5. Parallel Approximation Scheme for 0-1 Knapsack

Compute B such that $B \leq f(S_0) \leq 2B$
set subinterval size $l \leftarrow \frac{\epsilon B}{n}$

The following code is executed by each PE_i; $1 \leq i \leq \frac{2n}{\epsilon}$.
Each PE has three private variables *NewP*, *NewA*, and *NewI*.

$P[1] \leftarrow A[1] \leftarrow 0$ $\qquad$ $P[i] \leftarrow A[i] \leftarrow$ undefined for $1 < i \leq \frac{2n}{\epsilon} + 1$

FOR $j \leftarrow 1, 2, \ldots, n$

<*Synchronize*>

/* *Expand Phase* */
NewI ← undefined
IF A[*i*] is defined and $A[i] + a_j \leq b$ **THEN**
NewP ← $P[i] + p_j$
NewA ← $A[i] + a_j$
NewI ← $\lfloor NewP / l \rfloor$

/* *Compact Phase* */
IF *i* is odd and NewI is defined **THEN**
IF A[NewI] is undefined or A[NewI] > NewA **THEN**
P[NewI] ← NewP $\qquad$ A[NewI] ← NewA $\qquad$ (†)

<*Synchronize*>

IF *i* is even and NewI is defined **THEN**
IF A[NewI] is undefined or A[NewI] > NewA **THEN**
P[NewI] ← NewP $\qquad$ A[NewI] ← NewA $\qquad$ (††)

Theorem 9. Algorithm 5 has parallel time complexity $O(\frac{n^2}{p\epsilon})$ with $1 \leq p \leq \frac{2n}{\epsilon}$ PE's, and produces an ϵ-approximate solution for any $0 < \epsilon \leq 1$.

Proof. The bound B can be computed in $O(\log n)$ time using $O(n)$ PE's as explained in the proof of Theorem 7. The only difference is that the objects are sorted into non-increasing order by *profit density ratio*, $\frac{p_i}{a_i}$, instead of by weight, a_i. Each stage of Algorithm 5 clearly takes constant time when one PE is assigned to each subinterval, so the total time is $O(n)$ when $\frac{2n}{\epsilon}$ PE's are used. When there are fewer than $\frac{2n}{\epsilon}$ PE's, each PE is assigned to several subintervals. It is not hard to see that Algorithm 5 retains all of the partial solutions that Algorithm 4 retains, so the error analysis follows directly from the analysis of Algorithm 4. Algorithm 5 may eliminate some partial solutions at a later stage than Algorithm 4 but this will not affect the error analysis. ■

A divide-and-conquer technique similar to the technique in Algorithm 3 can be used with Algorithm 5 to get a sublinear time algorithm. At the end of Algorithm 6 (next page), the largest element of array R will be the value of an ϵ-approximate solution as shown below.

Theorem 10. Algorithm 6 has parallel time complexity $O(\log^2 n + \log n \log(\frac{1}{\epsilon}))$, uses $O(\frac{n^2}{\epsilon^2})$ PE's, and produces an ϵ-approximate solution for any $0 < \epsilon \leq 1$.

Proof. As in Algorithm 3, there are $\log n$ parallel divide-and-combine steps and at most $\frac{n}{2}$ subproblems are being solved in each step. For each subproblem, step 2 can be performed in constant time with $O(\frac{n^2}{\epsilon^2})$ PE's because each pair of arrays (Q_i, B_i) can be formed in constant time using $O(\frac{n}{\epsilon})$ PE's and there are at most

$O(\frac{n}{\epsilon})$ arrays. Two (pairs of) arrays of partial solutions can be combined in constant time in step 3 using $O(\frac{n}{\epsilon})$ PE's and the compact phase of Algorithm 5. Step 3 therefore takes $O(\log(\frac{n}{\epsilon}))$ parallel time if $O(\frac{n^2}{\epsilon^2})$ PE's are used and the entire algorithm is $O(\log n \log(\frac{n}{\epsilon})) = O(\log^2 n + \log n \log(\frac{1}{\epsilon}))$. The error analysis for this algorithm is straightforward. A partial solution can only be eliminated by another partial solution in the same subinterval and this introduces an absolute error of at most $\frac{\epsilon B}{n}$. Thus, the execution of step 3 introduces an absolute error of at most $\frac{\epsilon B}{n}$. When pairs of partial solutions are combined in step 2, the absolute error may double, but this can happen at most $\log n$ times. So, the total absolute error is at most ϵB and the relative error is at most ϵ. ■

Algorithm 6. Sublinear Parallel Approximation Scheme for 0-1 Knapsack

1. Divide the set of n items into two disjoint subsets of $\frac{n}{2}$ items each and solve the problem for each subset recursively and in parallel. The result will be two (pairs of) arrays of partial solutions, say (P_1, A_1), and (P_2, A_2). Each array contains at most $\frac{2n}{\epsilon} + 1$ elements.
2. For each partial solution $(P_1[i], A_1[i])$; $1 \leq i \leq \frac{2n}{\epsilon} + 1$, generate a pair of arrays $(Q_i, B_i) = \{(P_1[i] + P_2[j], A_1[i] + A_2[j]) \mid A_1[i] + A_2[j] \leq b;\ 1 \leq j \leq \frac{2n}{\epsilon} + 1\}$.
3. Recursively and in parallel, combine the arrays of partial solutions from step 2 using the compact phase of Algorithm 5 to get a single pair of arrays of partial solutions, say (R, C). These arrays will contain at most $\frac{2n}{\epsilon} + 1$ elements each and correspond to arrays P and A in Algorithm 5 at the end of a stage.

Clearly, Algorithm 6 can be used to solve the Subset Sum problem but it will be slower than Algorithm 3 when $\log n > \log(\frac{1}{\epsilon})$ and it uses more PE's. Algorithm 6 can also be used to solve the Unbounded Knapsack problem in which multiple copies of elements can be used in partial solutions.

Theorem 11. An ϵ-approximate solution for an instance of the Unbounded Knapsack problem can be found in $O(\log^2(\frac{n}{\epsilon}))$ parallel time with $O(\frac{n^2 \log^2 b}{\epsilon^2})$ PE's for any $0 < \epsilon \leq 1$.

Proof. Replace each element (p_i, a_i) of the input by the elements $(p_i, a_i), (2p_i, 2a_i), (4p_i, 4a_i), \ldots, (2^k p_i, 2^k a_i)$ where $k = \lfloor \log(\frac{b}{a_i}) \rfloor$ and solve the resulting 0-1 Knapsack problem. ■

The techniques used to derive the fast parallel approximation schemes in this section can be used to solve a variety of other problems. Some of these problems are Job Sequencing with Deadlines, Minimum Finish Time Scheduling, and Weighted Mean Flow Time Scheduling. The necessary modifications are not difficult and follow changes in the serial algorithms in [7], [13], and [14].

5. CONCLUDING REMARKS

Most of the algorithms in this paper introduce some overhead to achieve faster running times, so it may be possible to improve some of the bounds. Algorithm 2 for the Subset Sum problem is $O(\frac{n}{p\epsilon} \log(\frac{1}{\epsilon}))$ using $1 \leq p \leq \frac{8}{\epsilon}$ PE's and was derived from an $O(\frac{n}{\epsilon})$ serial algorithm. When $p = 1$, Algorithm 2 is $O(\frac{n}{\epsilon} \log(\frac{1}{\epsilon}))$, so an extra $O(\log(\frac{1}{\epsilon}))$ factor was introduced. Algorithm 5 for the 0-1 Knapsack problem is much better in this respect. It was derived from Algorithm 4 and is $O(\frac{n^2}{p\epsilon})$ using $1 \leq p \leq \frac{2n}{\epsilon}$ PE's. When $p = 1$, the bound is $O(\frac{n^2}{\epsilon})$, the same as the bound for Algorithm 4, so no overhead was introduced. The divide-and-conquer technique also introduces overhead. Algorithm 3 for the Subset Sum problem is

$O(\log n \log(\frac{1}{\epsilon}))$ using $O(\frac{n}{\epsilon^2})$ PE's. When only $O(\frac{1}{\epsilon})$ PE's are used, the running time becomes $O(\frac{n}{\epsilon}\log n \log(\frac{1}{\epsilon}))$. A comparison with the $O(n\log(\frac{1}{\epsilon}))$ bound for Algorithm 2 (using $O(\frac{1}{\epsilon})$ PE's) shows that the divide-and-conquer technique introduces an extra $O(\frac{\log n}{\epsilon})$ factor. A similar comparison of Algorithms 5 and 6 for the 0-1 Knapsack problem shows that an overhead factor of $O(\frac{\log n}{\epsilon}(\log n + \log(\frac{1}{\epsilon}))$ was introduced in Algorithm 6.

ACKNOWLEDGEMENTS

The authors would like to thank Larry Stockmeyer for suggesting the use of divide-and-conquer, and Stephen Cook for several ideas which improved the sublinear time bounds.

REFERENCES

[1] S. Fortune, J. Wyllie. "Parallelism in random access machines". *Proc. 10th Symp. on Theory of Computing*, May 1978, 114-118.

[2] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.

[3] G. V. Gens, E. V. Levner. "Approximate algorithms for certain universal problems in scheduling theory", *Izv. Akad. Nauk SSSR, Tech. Kibernet.* **16:6**(1978), 38-43.

[4] G. V. Gens, Ye. V. Levner. "Discrete optimization problems and efficient approximate algorithms (A survey)". *Eng. Cybernetics* **17:6**(1979), 1-11.

[5] G. V. Gens, E. V. Levner. "Fast approximation algorithms for knapsack type problems", *Proc. 9th IFIP Conf. on Optimization Techniques*, Sept. 1979. *Lecture Notes in Control and Information Sciences, Vol. 23*, Springer-Verlag, Berlin, 185-194.

[6] G. Gens, E. Levner. "Complexity of approximation algorithms for combinatorial problems: a survey", *SIGACT News* **12:3**(Fall 1980), 52-65.

[7] E. Horowitz, S. K. Sahni. "Exact and approximate algorithms for scheduling nonidentical processors", *JACM* **23:2**(Apr 76), 317-327.

[8] O. H. Ibarra, C. E. Kim. "Fast approximation algorithms for the knapsack and sum of subsets problems", *JACM* **22:4**(Oct 75), 463-468.

[9] E. L. Lawler. "Fast approximation algorithms for knapsack problems". *Proc. 18th Symp. on Foundations of Computer Science*, Nov. 1977, 206-213.

[10] T. Leighton. "Tight bounds on the complexity of parallel sorting". *Proc. 16th Symp. on Theory of Computing*, May 1984, 71-80.

[11] S. Moran. "General approximation algorithms for some arithmetical combinatorial problems", *Theor Comp. Sc.* **14:3**(June 81), 289-303.

[12] J. Peters, L. Rudolph. *Parallel approximation schemes for subset sum and knapsack problems*. Tech. Rep. CMU-CS-84-155, Dept. of Computer Science, Carnegie-Mellon U., Aug. 1984.

[13] S. K. Sahni. "Algorithms for scheduling independent tasks", *JACM* **23:1**(Jan 76), 116-127.

[14] S. Sahni. "General techniques for combinatorial approximation", *Operations Res.* **25:6**(Nov 77), 920-936.

O(LOG N) PARALLEL ALGORITHMS FOR BINARY TREE TRAVERSALS

K.V.S. RAMARAO
Dept. of Computer Science, University of Pittsburgh, Pittsburgh, Pa. 15260

Abstract

We present O(logn) parallel algorithms for traversals on binary trees - preorder, inorder and postorder using $O(n^2/\log n)$ processors. We show that there is a close relationship between this problem and that of finding the number of descendants D(v) of each node v in the tree. Then we give algorithms which, given D(v) for all v, compute the traversals with O(n) processors and only O(log n) time for preorder and $O(\log^2 n)$ time in the other cases.

1. INTRODUCTION

Binary trees have found an enormous spectrum of applications in widely varied areas in computer science. Construction, dynamic maintainence, balancing and traversals on binary trees have been extensively studied in the sequential domain. (Any standard book on data structures, such as [1], deals with these problems in detail.) Recently, there has been a great amount of interest in designing parallel algorithms for various problems, particularly those in graph theory [2,3,4,5,6]. This surge of activity is justifiably motivated by technological advancements which have made parallel computers with large number of processors a reality and led to the advancements in VLSI. Thus, our interest in parallel algorithms arises from at least two concerns: a) to design algorithms which explicitly exploit parallelism inherent in problems, so as to program on parallel computers, and b) to take a first step towards implementations on VLSI chips. The importance of binary trees is well-appreciated and requires no special emphasis.

In this paper, we present O([n/k] + logn) time parallel algorithms for pre, in, and postorder traversals on binary trees, using nk processors for $k > 0$. Thus, we can find these tours in O(logn) time if $O([n^2/\log n])$ processors are available. We take an interesting direction towards reducing the number of processors by providing parallel algorithms which would use only as many processors as are required for another closely related problem. The paper is organized as follows: We present the necessary background and relation with a closely related problem in Section 2. Section 3 gives the algorithms and their analysis. Section 4 presents certain improvements. Section 5 concludes the paper.

2. BACKGROUND AND CHARACTERIZATION

We consider the PRAM model of computation where processors can simultaneously read a memory location but are not allowed to write.

A (rooted) tree is an acylic graph G = (V,E) with a special node r. It is a binary tree if no node has more than two sons. Assume that each node has an identity and that each node v in V knows f(v), left (v) and right (v) representing father of v, left son of v and right son of v respectively. If any of them do not exist, then the value of the corresponding variable is 0. Any node u lying on the path from r to v is called an 'ancestor' of v. Three kinds of traversals are defined recursively on a binary tree, as follows;

Preorder traversal:
1) Visit the root,
2) Make preorder traversal on the left subtree,
3) Make preorder traversal on the right subtree.

Inorder traversal:
1) Make inorder traversal on the left subtree,
2) Visit the root,
3) Make inorder traversal on the right subtree.

Postorder traversal: 1) Make postorder traversal on the left subtree.
2) Make postorder traversal on the right subtree.
3) Visit the root.

Each of the traversals can be performed in O(n) sequential time. Straightforward parallel simulations of the sequential algorithms take O(d) time where d is the depth of the tree, which can be O(n) in the worst case. To do any better, we need to find the position of each node in a 'tour' (the sequence in which the nodes are visited during a traversal) independent of all other nodes.

Let D(v) be the number of nodes in the subtree rooted at v (including v). The following result suggests an interesting approach to computing the tours.

Lemma 1. Any sequential algorithm A which generates the preorder, inorder, or postorder tour on a binary tree can compute D(v) for all v in V in O(n) more time.

Proof: Following algorithms achieve this task:

a) Let the array V[1..n] represent the preorder tour. *We find the preorder number of each node*

```
for  i:= 1 to n do
Prenum (V(i)) := i
od
*Next (i) is the delimiter for the subtree at i th node in the tour*
Next (r) := 0 ;
for  i := 2 to n do
  if right (f(V(i))) ≠ 0 AND right (f(V(i))) ≠  V(i) then
        Next (V(i)) := right (f(V(i)))
   else  Next (V(i)) := Next (V(i-1))
  fi
od
*D(i) can be computed now*
for i  := 1 to n do
if Next (V(i)) = 0 then D(i) := n - Prenum (i)
    else D(i)  := Prenum (Next (i)) - Prenum (i)
  fi
od
```

b) Inorder traversal: Finding the subtree is more complicated here since, in the inorder tour, the first node in the subtree at v is not v. Hence, we need to find the inorder numbers of the first and last nodes in the subtree at v.

```
*computing the first node*
    start (r) :=V(1);
    Bucket Sort the nodes on their depth (that is, length of the path from
the root to a node);
    Let d [1..n] be the array obtained.
    for i := 2 to n do
    if left (f(d(i))) = i then start (d(i)) := start (f(d(i)))
      else start (d(i))  := V(Innum (f(d(i))) + 1)
    fi
    od
*computing the last node*
for i := 1 to n do
if V(i) = r then Next (r) := 0
else
if  left (f(V(i))) = V(i) then Next (V(i)) := f(V(i))
    else Next (V(i)) := Next (f(V(i)))
fi fi
od
```

c) Postorder traversal:

Computing the first node in a subtree is same as above.

```
 * computing the last node *
   for  i := 1 to n do
   if V(i) = r then Next (r) :=0
     else
       if left (f(V(i)) = V(i) then
         if right (f(V(i))) = 0 then Next (V(i)) := f(V(i))
           else Next (V(i)) := start (right(f(V(i))))
         fi
     else Next (V(i)) := f(V(i))
     fi
   fi
   od
```

Correctness of these algorithms follows from the definitions and each of them clearly take O(n) time QED

Similar results can be obtained in the parallel domain. Here, each node finds the first and last nodes in its subtree concurrently. This concurrency seems to increase the time by a log n factor as it happens for several problems. First, we present two known fundamental results:

Lemma 2. [3] Given n elements $\{a_0, a_1, \ldots, a_{n-1}\}$ partitioned into p sets and k processors, products for each of these sets under operation * can be performed in T parallel time units where * is any associative binary operator and

$$T = [n/k] - 1 + \log k \text{ if } [n/2] > k,$$
$$= \log n \text{ if } [n/2] \leq k.$$

Another important result we use is as follows:

Lemma 3. [7] Given are n elements $\{a_0, a_1, \ldots, a_{n-1}\}$. Let f be a function to be applied to every element. If computing $f(a_i)$ takes t time units and k processors are provided, then $f(a_i), 0 \leq i \leq n-1$, can be computed in [n/k].t parallel time units

Theorem 1. Any parallel algorithm A solving the binary tree traversal problem using k processers can determine D(v) for all v in no more than O([n/k]log n + log n) additional time.

Proof: As remarked above, we determine the first and last nodes in the subtree at each node. Here, we present an algorithm for the preorder traversal. Other two are similar.

```
   for all v in V pardo
 * we find last (v) in log2n steps *
   if right (v) ≠ 0 then L^1(v) :=right (v)
     else if left(v) ≠ 0 then L^1(v) := left(v)
                 else L^1(v) := v
          fi
   fi
 for k := 1 to [log2n] do
     L^(2^k) (v) := L^(2^(k-1)) (L^(2^(k-1)) (v))
 od
     D(v)  :=  Prenum (L^(2^[log n]) (v))- Prenum (v) +1
 dopar
```

It is clear from Lemma 2 that this can be done with the resources claimed. QED

We attack the problem of finding the tours by computing D(v) first.

By the above result, this is justifiable as long as the complexity of generating the tours using this information is no more than that of computing D(v)'s.

Definition 1. [4] For each v in V, $F^0(v) := v$;

$F^1(v) := f(v)$ and $F^k(v) := F^{k-1}(f(v))$.

Intuitively, $F^k(v)$ gives the k the ancestor of v. By definition, $F^k(r) = r$ for all k.

Definition 2. [5] For $1 \leq i, j \leq n$ such that depth(i) $\geq$ depth(j), $F^+(i,j) = F^{\text{depth}(i)-\text{depth}(j)}(i)$.

Following is a known result:

Theorem 2. [5]. The function F^+ can be computed in $O([n/k]+\log n)$ time using $O(nk)$ processors, for $k > 0$.

This result is optimal since F^+ can be defined for $O(n^2)$ instances in the worst case (for skewed trees).

Theorem 3. Given F^+, D(i), $1 \leq i \leq n$, can be computed in $O([n/k] + \log n)$ time using $O(nk)$ processors, for $k > 0$.

Proof: For each j, we count the number of times j appears in the (n-1)-depth (j) th 'column' in the table F^+. That is, we compute the number of times j occurs among F^+ (i, (n-1) - depth (j)), $1 \leq i \leq n$. This is clearly D(j). By Lemma 2, this can be done with the given resources. QED.

Note that we have not used the fact that the tree is binary until now. Since D(v) can be found in O(n) sequential time, the above algorithm is not optimal. Now, we present algorithms to compute the pre, in and postorder tours on a binary tree. We take advantage of the fact that the function F^+ is precomputed.

Algorithm Preorder:

for v in V pardo

Step 1. For each ancestor u of v, set left-sub (v,u)=1 if v is in the left subtree of u and 0 otherwise.

Step 2. if left-sub(v,u)=1 then con(v,u) := 1
else con(v,u) := D(left(u)) + 1
fi

Step 3. Prenum (v) := Sum {con(v,u) | u is a proper ancestor of v}

dopar

Prenum (r) := 1 ;

Algorithm Inorder:

for all v in V pardo

Step 1. Same as above.

Step 2. if left-sub(v,u) =1 then con (v,u) := 0
else con(v,u) := D(left(u)) + 1
fi

Step 3. Innum (v) := Sum {con(v,u) | u is a proper ancestor of v}

Algorithm Postorder:

for all v in V pardo

Step 1. As above.

Step 2. if left-sub(v,u) = 1 then con(v,u) := 0
else con(v,u) := D(left(u))
fi

Step 3. Postnum(v) := Sum {con(v,u) | u is a proper ancestor of v}

Lemma 4. Step 1 in the above algorithms can be computed in O([n/k] + logn) time with O(nk) processors, for k>0.

Proof: Following is an algorithm to implement Step 1. This is similar to the algorithm for generating F^+ [5].

begin

1. for i : $1 \leq i \leq n$ pardo
if left (f(i)) = i then $G^1(i) := 1$ else $G^1(i) := 0$
fi
dopar

2. for t :=0 to log(n-1)-1 do
for s : $1 \leq s \leq 2^t$, i : $1 \leq i \leq n$ pardo
$G^{2^t+s}(i) := G^{2^t}(G^s(i))$
dopar
od

3. Right Shift each 'row' so that all the occurrences of root except the leftmost one are eliminated. The resulting function is left-sub.

end

As in [5], it can be shown that this algorithm can be implemented within the given resource bounds. QED.

Lemma 5. Steps 2 and 3 in the above algorithms can be implemented in O([n/k] + logn) time using O(nk) processors, for k >0.

Proof: We apply Lemma 3 for Step 2. The function f is 'if..then..else' which takes a unit time and the number of elements is $O(n^2)$. Thus, Step 2 takes O([n/k]) time. Lemma 2 can be applied to Step 3 to give the required bound. QED

From Lemmas 4 and 5, we have,

Theorem 4. Each of the algorithms Preorder, Inorder and Postorder can be implemented in O([n/k] + logn) time with O(nk) processors, for k >0.

Theorem 5. Each of the above algorithms correctly computes the respective tours.

Proof: The variable 'con' in the algorithms gives the contribution made by each of the ancestors of a node v towards its position in the corresponding traversal. It can be checked from the definitions of traversals that these contributions are correctly given. QED

4. POSSIBLE IMPROVEMENTS.

In the previous section, we have noted that the algorithm used to compute D(v) is suboptimal. Furthermore, we have explicitly made use of the function F^+ in computing the tours. Since the algorithm for F^+ is optimal, no improvements to the performance of our algorithms are possible unless the usage of F^+ is eliminated. Observe that it is not necessary to have F^+ to compute D(v), though we do not currently have any better scheme. In this section, we show that we can eliminate using

F^+, so that the complexity of our algorithms depends on that of finding D(v). Following simple result provides the basis for this in case of preorder traversal:

Lemma 6. Let u be a proper ancestor of v. Then Prenum (u)< Prenum (v).

This 'monotonicity' of preorder makes it possible to use the information of no more than [log n] proper ancestors of v in computing the preorder number. Here is an algorithm for preorder traversal using this idea:

Algorithm Mod-Preorder:

for all v in V pardo

Step 1. For $0 \leq k \leq [\log n]$, set left-sub$^{2^k}$ (v) := 1 if v is in the left subtree of F^{2^k} (v) and 0 otherwise.

Step 2. if left(f(v)) = v then
Prenum^1(v) := 2 else Prenum^1(v) := 2+D(left(f(v)))
fi

Step 3. for k := 1 to [log n] do
if F^{2^k} (v) = $F^{2^{k-1}}$(v) then exit fi
Temp := Prenum$^{2^{k-1}}$ (v) + Prenum$^{2^{k-1}}$ ($F^{2^{k-1}}$ (v))-1;
if left-sub$^{2^k}$ (v) = 1 then Prenum$^{2^k}$ (v) := Temp
else Prenum$^{2^k}$ (v) := Temp+D(left($F^{2^{k-1}}$ (v)))
fi
od

dopar

Correctness of this algorithm can be proved by induction on k.

Theorem 6. The algorithm Mod-Preorder finds the preorder tour in O([n/k] log n + log n) steps when k> 0 processors are available and D(v) is given for all v in V.

Proof. Each of the for loops is executed O(log n) times and the instructions in each such loop take only a constant time. Hence, using Lemmas 2 and 3, the algorithm can be implemented with the given resources. QED

Thus, if D(v) can be computed using less than $O([n^2/\log n])$ processors, so is the preorder tour.

For inorder and postorder traversals, we observe that, for any node v, contributions in determining its position in the tour are made only by those ancestors for which v is in the right subtree. We use this fact to design algorithms which do not use F^+.

Algorithm Mod-Inorder:

for all v in V pardo

Step 1. R^1(v) := nearest ancestor u of v such that v is in the right subtree of u

Step 2. if R^1(v) = 0 then In^1(v) := D(left(v)) + 1 else
In^1(v) := D(left(R^1(v))) + 2 + D (left(v))
fi

Step 3. for k := 1 to [log n] do

$R^{2^k}(v) := R^{2^{k-1}}(R^{2^{k-1}}(v))$

if $R^{2^k}(v) = R^{2^{k-1}}(v)$ then exit fi

$In^{2^k}(v) := In^{2^{k-1}}(v) + In^{2^{k-1}}(R^{2^{k-1}}(v)) - 1$

od

Step 4. $Innum(v) := In^{2^{[\log n]}}(v)$

dopar

Algorithm Mod-Postorder

for all v in V pardo

Step 1. $R^1(v)$:= nearest ancestor u of v such that v is in the right subtree of u

Step 2. If $R^1(v) = 0$ then $Post^1(v) := 0$ else $Post^1(v) := D(left(R^1(v)))$

fi

Step 3. for k := 1 to [log n] do

$R^{2^k}(v) := R^{2^{k-1}}(R^{2^{k-1}}(v))$

if $R^{2^k}(v) : R^{2^{k-1}}(v)$ then exit fi

$Post^{2^k}(v) := Post^{2^{k-1}}(v) + Post^{2^{k-1}}(R^{2^{k-1}}(v))$

od

Step 4. $Postnum(v) := Post^{2^{[\log n]}}(v) + D(v)$

dopar

Theorem 7. Algorithms Mod-Inorder and Mod-Postorder generate inorder and postorder tours in $O([n \log^2 n/k] + \log^2 n)$ parallel time using nk processors, when D(v) is given for all v.

Proof Consider Mod-Inorder, Mod-Postorder being similar. Firstly, the correctness follows from the remark made before the algorithm Mod-Inorder. Steps 2 and 4 clearly take constant time while Step 3 takes O(log n) time. Step 1 can be easily shown to take $O(\log^2 n)$ time. Following is an algorithm:

for all v in V pardo

if left (f(v)) = v then $H^1(v) := 0$ else $H^1(v) := 1$

fi

for k := 1 to [log n] do

$H^{2^k}(v) := H^{2^{k-1}}(F^{2^{k-1}}(v)) + H^{2^{k-1}}(v)$

od

dopar

Observe that the sequence $H^1(v), H^2(v), \ldots, H^{2^{[\log n]}}(v)$ is sorted in ascending order, and, if $H^{2^k}(v) > 0$ for some k, then $R^1(v)$ is a descendent of $F^{2^k}(v)$. Thus, we make a binary search on H to find the

smallest k such that $H^{2^k}(v) \geq 0$. We can now perform a binary search on the array $H^1(F^{2^{k-1}}(v),\ldots,H^{2^{\lceil \log n \rceil}}(F^{2^{k-1}}(v))$, and so on until we find $R^1(v)$. Clearly, this process takes no more than $O(\log^2 n)$ units of time. By Lemma 3, the resources claimed are sufficient. QED

5. CONCLUSIONS.

We have shown that the parallel traversal on binary trees is at least as hard as that of finding D(v) for all v. Conversely, algorithms are presented for traversals which use only D(v) and take no more than $O(\log^2 n)$ time with O(n) processors. Since the existing algorithms for finding D(v) use $O(n^2/\log n)$ processors to achieve O(log n) time bound, we have given algorithms for traversals also which use the same resources. The central problem in designing parallel algorithms for problems on trees seems to be that of finding D(v). If better algorithms are found for this problem, the performance of our traversal algorithms can also be improved.

REFERENCES

[1] D.E. Knuth, Art of Computer Programming, Vol. 1: Fundamental Algorithms, Addison-Wesley, Reading, Ma., 2nd Ed., 1973.

[2] E. Reghbati and D.G. Cornell, "Parallel Computations in Graph Theory," SICOMP, VOL. 7, No. 2, May 78, pp. 230-237.

[3] F.Y. Chin, J. Lam and I-Ngo Chen, "Efficient Parallel Algorithms for Some Graph Problems," CACM Vol. 25, No. 9, Sept. 82, pp. 659-665.

[4] C.D. Savage and J.Ja'Ja', "Fast, Efficient Parallel Algorithms for Some Graph Problems," SICOMP, Vol. 10, No. 4, Nov. 81, pp. 682-691.

[5] Y.H. Tsin and F.Y. Chin, "Efficient Parallel Algorithms for a class of Graph Theoritic Problems," to appear in SICOMP.

[6] J. Ja'Ja' and J. Simon, "Parallel Algorithms in Graph Theory: Planarity Testing," SICOMP, vol. 11, No. 2, May 82, pp. 86-90.

[7] Y.H. Tsin, "Portable Efficient Algorithms for Graph Theoritic Problems," Ph.D. Thesis, University of Alberta, Edmonton, Canada, Fall 1983.

SIMPLE AND EFFICIENT DISTRIBUTED ALGORITHMS FOR ELECTION IN COMPLETE NETWORKS

YEHUDA AFEK
ELI GAFNI
Computer Science Department
University of California, Los Angeles, CA 90024

Abstract

The message complexity of an election algorithm in a general network is $O(m+n\cdot\log n)$. Recently, an $O(n\log n)$ election algorithm for a complete network was presented [6]. However, the algorithm is quite complicated in terms of understanding and programming. Here, we present simple and short election algorithms which are as efficient. The suggested algorithms are of practical importance for users who are actually interested in programming the election algorithm on a complete network of processors.

1 Introduction

The election algorithm is an identical program residing in every node in the network. Initially, nodes differ only by their identifiers (*ids*). The nodes elect a leader by exchanging messages. When the message exchange terminates, one node is distinguished from all others.

In [5], an $O(m+n\cdot\log n)$ election algorithm for general networks was presented where m is the number of links and, n is the number of nodes in the network. It was later proven in [1,7] that $O(n\cdot\log n)$ is a lower bound. A lower bound of $O(m)$ is obvious since any untraversed link could be the only link connecting two parts of the network, each holding a separate election algorithm.

In, [6] Korach et. al. showed that the $O(m)$ lower bound does not hold for the special case of a complete network, i.e., when every node has a link to every other node in the network. The authors then presented an election algorithm for complete networks. The algorithm follows that presented in [5] and, in terms of understanding and programming, is as complex.

In complete networks, an election algorithm may terminate if all the the neighbors of one node have been visited. This is not the case in the general network. Taking advantage of the simplicity of termination detection, we apply principles suggested in [4,5] for general networks to derive two simple and elegant algorithms (A and B) for election in complete networks. The two algorithms present a tradeoff between time and message complexities. Algorithm A has an $O(n)$ time complexity and $2.773\cdot n\cdot\log n$ message complexity. Algorithm B has a $2\cdot n\cdot\log n$ message complexity but an $O(n\cdot\log n)$ time complexity. Analyzing the communication and time complexities of the two algorithms, we

*This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-82-C-0064. The first author is an IBM Graduate Fellow.

derive a third algorithm, algorithm C, which achieves the time complexity of algorithm A and the communication complexity of algorithm B. The principles in [4] have already been successfully applied in [2] to produce an efficient election algorithm in unidirectional networks. These results suggest that the principles behind the presented algorithms are fundamental to the area of distributed algorithms.

2 Algorithms for Election in Complete Networks

2.1 Overview

We present three algorithms (A, B, and C) for election in a complete network. The underlying mechanism for all three algorithms is similar. Each algorithm is initiated by any subset of nodes, each of which spawns a process. Each spawned process tries to capture all the other nodes by successfully traversing in both directions all the links incident to its initiator, called the *base*. The *base* node whose process has succeeded in capturing all its neighbors, elects itself as the leader of the network. To guarantee that only one node is elected, all processes but one are *killed*.

Processes in the different versions of the algorithm use a combination of two variables, *id* and *level*, in order to contest each other. The process *id* is the id of its *base* node. The process *level* is an estimate of the amount of work it has already done. Process *level* could, for example, be some function of the number of nodes it has captured, the number of processes it has killed or both.

To keep track of its owner process, every captured node has two link pointers, *father* and *potential-father*. In general, the *father* pointer points to the link through which the node was most recently captured. The *potential-father* pointer points to the link through which a process which tries to capture the node has arrived.

Every captured node has a *level* variable, which is the highest level process it has observed.

A process that arrives at a node with a larger *level* than its own, kills itself. On the other hand, if the node's *level* is smaller or equal to that of the process, the node's *level* is replaced by the process *level*, which is then said to survive. The surviving process may then try to kill the previous owner of the node. The different criterion which the process uses to kill the previous owner give rise to the different algorithms.

The simplicity of the election algorithms stems from the simplicity of termination detection and process synchronization in a complete network. Termination is easily detected by a *base* node when its process has captured all its neighbors. Synchronization between two conflicting processes is easy since each captured node is at most one hop away from its owning *base* node.

The main difference between the three algorithms is the way that processes determine their *level*. In algorithm A, processes use the number of nodes they have captured as their *level*. In algorithm B, processes use the number of processes they have killed. Among the two algorithms, algorithm A achieves a better time complexity while

algorithm B achieves a better communication complexity. In algorithm C, processes use a combination of the above two *level* functions to achieve the message complexity of algorithm B and the time complexity of algorithm A.

2.2 Algorithm A

In this algorithm, the *level* of a process is the number of nodes it has already captured. When a process replaces the *level* of an already captured node, it has to kill the previous owner of the node before it may claim the node to itself.

To resolve ties, the *process id* (*base id*) is appended to the *level* of every process and every node, and comparisons are carried out lexicographically. Nodes initialize their *level* to 0. Processes initialize their *level* to 0 appended by their *base-id*.

A process that arrives at an already captured node v whose *level* is smaller than its own, replaces v's *level* with its own and becomes v's *potential-father*. The *potential-father* process is then sent to the *base* node which owns v. If the process survives at v's *base* node, and no other process became v's *potential-father* in the meanwhile, then it becomes v's *father*.

Figure 1 shows the formal description of algorithm A. In the algorithm, E is the set of edges incident to a node. Every *base* node maintains a list of edges, called *untraversed*, which it's process has not yet traversed in any direction. The untraversed list is used by *base* nodes, live and dead, to decide whether a visiting process is one which tries to capture it or is a conflicting *potential-father*. A process which comes over a link from the untraversed set is a capturing process while a process which arrives through an already traversed link is certainly a disputing *potential-father*. In this way the algorithm has only one type of message.

The algorithm is deadlock free since processes never wait for each other, and the (*level*, *id*) pair is lexicographically increasing along any chain of processes which kill each other.

The time complexity of the algorithm is $O(n)$ since processes never wait for each other and a process which has done more work is never killed by a process which has done less work. Thus, each killed process spent less time, in the worst case, than the process killing it.

The analysis of the communication complexity of the algorithm relies on the following observation: The sets of captured nodes, each owned by a different live process, are disjoint. This is because every node has at most one *father*, and when a node changes a *father*, the previous one is dead. Hence, if one process owns the entire network, it is the only live process in the network.

To prove that the communication complexity of the algorithm is $O(n \cdot \log n)$ we use a Lemma which was introduced in [4].

```
level := owner-id := 0 ;
untraversed := E ; father := nil ;

     /* The following is performed by initiator nodes */

Base ( id ) :
while ( untraversed ≠ ∅ ) do;
      e := any( untraversed ) ;
      send(id,level) on e ;
R:    receive(id',level') over e' ;
      if (id' = id)  then          /*process returns from successful capturing.*/
           level := level + 1 ;
           untraversed := untraversed - e ;
      else                         /*another process tries to capture the node.*/
          if (level',id' < level,id)      /* comparison done lexicographically*/
          then  Discard the message, goto R ;
          else  goto Killed ;
end while;

announce(ELECTED,terminate the algorithm) , STOP ;     /* untraversed = ∅ */

     /* The following is performed by non-initiator nodes */
Captured:
for_ever do;
    receive(id',level') over e' ;
    case level', id' of :
    (1) level',id' < level,owner-id:
          Discard message ;
    (2) level',id' > level,owner-id :
Killed :  potential-father := e' ;
          level := level' ;
          owner-id := id' ;
          /*Now check if id' is a disputing potential-father or a capturing process*/
          if (e' ∈ untraversed) then send(id',level') over e', goto Captured ;
          if father = nil then father:=potential-father ;
          send(id',level') over the father link ;
    (3) level', id' = level,owner-id:    /* Potential father has killed the father */
          father := potential-father ;
          send(id', level') over the father link ; /* Return the process to it's base.*/
    end case ;
end for_ever ;
```

FIGURE 1: Algorithm A

Lemma 1: For any given k, the number of processes that own $\frac{n}{k}$ nodes or more is, at most, k.

The proof of Lemma 1 relies on the disjointness property. Details can be found in [4].

Corollary 2: The largest process to be killed by another process owns at most $\frac{n}{2}$ nodes, the next largest at most $\frac{n}{3}$, etc.

Since to capture a node a process makes at most 4 hops, we arrive at the following:

Lemma 3: The total cost of algorithm A is $4 \cdot n \cdot \ln n$.

proof: The cost incurred by a process of size k is $4 \cdot k$ messages. From corollary 2, the total cost is bounded by $4 \cdot n \cdot \sum_{i=1}^{n} \frac{1}{i}$ messages. Hence, the total message complexity of the algorithm is bounded by $4 \cdot n \cdot \ln n$ ($2.773 \cdot n \cdot \log n$) messages. ■

Each message of algorithm A contains at most $2 \cdot \log n$ bits.

The principle behind algorithm A is that the node sets owned by different live processes are disjoint. Thus, a process has to verify that the father of every node it tries to capture is dead before claiming the node to itself. This mechanism causes a process, which captures many nodes from another process, to "kill" that other process as many times as the number of nodes it captures from it and hence, causing the factor 4 in the message complexity. In the next algorithm we relax the disjointness requirement, thus enabling a process to claim a node without killing its previous owner. This modification reduces the message complexity to $2 \cdot n \log n$ messages ($2.885 \cdot n \cdot \ln n$).

2.3 Algorithm B

As in the previous algorithm, the *level* of a process is initialized to 0; however, here we do not append the process id to the process *level*. Whenever a process sees another one with a higher level, it kills itself. If a process sees another one at the same *level* but with a smaller id, it kills the other process and increases its *level* by one. Thus, the *level* of a process is the number of processes it has killed. As in algorithm A, the node's *level* is the highest *level*-process it has seen.

Unlike algorithm A, processes in this algorithm may claim a node of another process without killing the other process. However, a process may replace the *level* of a node only if the node *level* was less than the process *level*, in which case the father pointer of the node is set to the link through which the process arrived.

If the process *level* equals node v's *level* and its id is larger than that of v's owning process, then the process becomes v's *potential-father*. The *potential-father* is then sent to v's father, f, in an attempt to kill it. If another process in the same *level* with an even higher id arrives at v before the *potential-father* returns from f, then this other process is killed. If the *potential-father* survives at f it returns to its *base* (via v which it now owns) and increments its *level* by one. However, if the *potential-father* process arrives at node f and f was killed already, then the *potential-father* kills itself.

```
/* The variable size is for algorithm C only */

      level := size := 0 ; owner-id := potential-id := 0 ;
      untraversed := E ; father := potential-father := nil ;

/* The following is performed by initiator nodes */

Base ( id ) :
while ( untraversed ≠ ∅ ) do;
      e := any( untraversed ) ;
      send(level,id) on e ;
R:    receive(level',id') over e' ;
      if (id' = id) then          /*process returns from successful capturing.*/
            level := level'
            untraversed := untraversed - e ;
      else-if (level',id' < level,id) then /* comparison done lexicographically*/
            Discard the message, goto R ;
      else-if e' ∈ untraversed then goto Killed ; /* capturing process from base id'*/
      else send(level',id') over e'; goto Captured ;/*disputing poten.-father process*/
end while;

announce(ELECTED, terminate the algorithm ), STOP ;   /* untraversed = ∅ */

/* The following is performed by non-initiator nodes */

Captured:
while (not terminated) do;
     receive(level',id') over e' ;
     if (e' ∉ untraversed) then Discard message ; /* cannot kill a dead process */
     else
Killed : case level' of :
       (1) level' < level : Discard message ;
       (2) level' > level :    /* Replace the father */
             father := e' ; level := level' ; owner-id := id' ;
             potential-id := 0; potential-father := nil ;
             send(level',id') over the father link ;
       (3) level' = level :
             if      (owner-id = 0) then /* id' becomes the father */
                   father := e' ; level := level' ; owner-id := id' ;
                   potential-id := 0; potential-father := nil ;
                   send(level',id') over the father link ;
             else-if (id' < owner-id) then Discard message ;
             else-if ( id' = potential-id ) then
                   father := potential-father ; level' := level' +1 ;
                   owner-id := id' ; potential-id := 0 ; potential-father := nil ;
                   send(level',id') over the father link ;
             else-if there is already a potential-father then Discard message ;
             else  /* there is no potential-father */
                   potential-id := id' ; potential-father := e' ;
                   send(level',id') over the father link ;
       end case ;
end while ;
```

FIGURE 2: Algorithm B

To analyze the performance of the algorithm we note that the maximum *level* achievable during its execution is $\log n$. The reason being, that at most half of the processes at level k can go up to level $k+1$. Clearly, every time a node is recaptured its level is increased by at least one. Hence, the total number of captures possible, is at most $n \cdot \log n$. Each capture costs 2 messages which sums to a total of $2 \cdot n \cdot \log n$. The extra message cost incurred by processes which go over *father* links to other *bases* is at most $2 \cdot n$, since each such traversal results in the elimination of one live process. Thus, the total message complexity of the algorithm is bounded by $2 \cdot n \cdot \log n + 2 \cdot n$ and the message length is $\log n + \log\log n$.

The time complexity of the algorithm is $O(n \cdot \log n)$ as can be seen from the following scenario. We will show how it is possible for $\frac{n}{2}$ of the nodes to be captured $\log\frac{n}{2}$ times. The algorithm is started by node, v_0 which captures $\frac{n}{2}$ nodes and is still in level 0. Then, a new node, v_1, spontaneously starts the algorithm, kills v_0, increases its level to 1 and captures the $\frac{n}{2}$ nodes. After v_1 has captured the $\frac{n}{2}$ nodes, two new nodes spontaneously start the algorithm, try to kill each other and the one which survives, v_2, reaches level 1. Node v_2 then kills v_1 and recaptures the $\frac{n}{2}$ nodes at level 2. The scenario continues until node $v_{\log\frac{n}{2}}$ recaptures the entire network and is elected as a leader.

The reason for the high time complexity of algorithm B is that killings are independent of the number of nodes captured by each of the disputing processes. Hence, a process which spent a lot of work (and time) accumulating nodes might be killed by a process which did not spend nearly as much. Although algorithm A does not suffer from this problem, it had the problem that processes could be "killed" many times. In the next algorithm we eliminate both problems by employing both techniques simultaneously in one algorithm.

2.4 Algorithm C

Here we make two modifications to algorithm B in order to achieve a linear time complexity with no increase in the communication cost. First, we combine an estimate of the amount of work spent by each process into the level function of algorithm B. Second, we enable processes with high levels ($>\log n$) to capture many nodes in parallel (in one time unit). We start describing the algorithm with the first modification. The second modification will be introduced during the performance analysis.

As before, the *level* of any process or node is initialized to 0. Processes increase their *level* according to two rules. First, as in algorithm B, whenever a process kills another one, it increases its *level* by 1. Second, when the process returns from a successful capturing, the *base* node replaces its *level* with the maximum of *level* and the log of the number of nodes it has already captured.

The algorithm is similar to algorithm B. In this algorithm *base* nodes use a *size* variable to count the number of nodes they have captured. *Base* nodes increase their *size* by one for each node captured by their processes and update their process *level* by *Max*(*level*,log*size*).

The formal description of the algorithm is similar to that of algorithm B. The only change is, to replace the first then clause, within the while loop of a *base* node program to:

then
size:= *size*+1 ;
level:= max(*level'*,log*size*) ;
untraversed := untraversed - e ;

To analyze its performances we will first show that:

Lemma 4: The maximum *level* reachable during any execution of algorithm C is $\log n + \log\log n + 1$.

proof: Let N_i be the total number of processes that reach *level i* during the execution of the algorithm. Consider the maximum number of processes which could possibly pass from *level i*-1 to *level i*. There are two ways in which a process can grow from *level i*-1 to *level i*. First, by capturing 2^{i-1} nodes at *level i*-1, for $i \leq \log n$ and second, by killing another process which is at *level i*-1. We note that N_i is maximized if as many processes as possible pass from *level i*-1 to *level i* by capturing other nodes (i.e., $\frac{n}{2^{i-1}}$) and the rest of the processes (i.e, $N_{i-1} - \frac{n}{2^{i-1}}$) kill each other in pairs. Hence,

$$N_i \leq \frac{(N_{i-1} - \frac{n}{2^{i-1}})}{2} + \frac{n}{2^{i-1}} \qquad (1)$$

Solving (1) for N_i we get:

$$N_i \leq \frac{n}{2^i} \cdot (i+1) \qquad (2)$$

Substituting $N_i = 1$ in (2) and solving for i gives us the maximum *level* which is, $\log n + \log\log n + 1$. ■

Using exactly the same argument as we used for the message complexity of algorithm B we get that the total number of messages of algorithm C is bounded by $2 \cdot n \cdot (\log n + \log\log n + 2)$ where each message contains at most $\log n + \log(\log n + \log\log n)$ bits.

With the above modification it can be shown that the time complexity of algorithm B was reduced to $O(n \cdot \log\log n)$ without any degradation in the big O message complexity. In order to further reduce the time complexity to $O(n)$, processes at level higher than $\log n$ will try to capture $\frac{n}{\log n}$ nodes in parallel. Thus a process which has reached *level* $\log n$ will send messages over $\frac{n}{\log n}$ untraversed links incident to its *base*. Each of these messages carry the (*level*, *id*) of the *base* node. When arriving at the adjacent nodes they compare their *level* to that of the nodes. If the message *level* is higher it replaces the (*level*, *id*) of the node thus becoming the father of the node and returning to the *base* notifying it of a successful capture. If the message *level* is smaller, it returns no message. Finally, if the message *level* is the same as that of the node but the message id is higher, a notification to that effect is sent back to the *base*.

The *base* node waits for all the $\frac{n}{\log n}$ messages. If all the messages indicate a successful capture, the *base* proceeds to the next $\frac{n}{\log n}$ untraversed incident links. If on the other hand, some of them indicate that they have encountered the same *level*, one of them is arbitrarily chosen and a process that behaves as in algorithm B is sent along the link. If the process return, the *base* node increases its *level* and proceeds to the next $\frac{n}{\log n}$ untraversed links (links on which no successful capture was reported are not considered traversed).

To analyze the algorithm with this modification we make two observations: First, the maximum attainable level in the algorithm is still bounded by $\log n + \log\log n + 1$. Second, by substituting $i = \log n$ in equation (2), we find out that the maximum number of processes which reach level $\log n$ is $\log n$.

The last modification has increased the communication complexity of the algorithm by at most $O(n)$ messages. Each node is still captured at most $\log n + \log\log n + 1$ times, however the death of a processes at level greater than $\log n$ might be associated with at most $2 \cdot \frac{n}{\log n}$ messages. Since there are at most $\log n$ such processes the increase due to killings is bounded by $O(n)$.

To show that the time complexity of the algorithm is $O(n)$ we arrange the processes in a rooted tree. Each level of the tree corresponds to the processes that have reached that *level* in the algorithm, i.e., the nodes at *level* i in the tree correspond to the processes that have reached *level* i in the algorithm. The parent of a process at *level* i in the tree is either the process that caused the death of the given process or, the same process at the next *level*. The time delay of the algorithm is the sum of the delays incurred by processes along the path from the first process spontaneously waking up, at *level* 0, to the root.

To evaluate this time delay we note that no process that either survives or is killed at the i^{th} *level*, spends more than 2^{i-1} time units in the i^{th} *level* for $i \leq \log n$. In levels higher than $\log n$ no process spends more than $\log n$ time units since it captures nodes at a rate of $\frac{n}{\log n}$ per time unit. Hence, the total time delay of the algorithm is bounded

by $\sum_{i=1}^{\log n} 2^i + \sum_{i=\log n}^{\log n+\log\log n} \log n = n+\log n \cdot \log\log n$. Note that we scale a time unit to be the maximum delay it takes to capture one node, which is a constant.

In the above calculation of the algorithm's time delay we did not include the actual time it takes processes to kill each other. Since, there are, at most, n processes and, no process tries to kill a dead one (unlike algorithm A), this delay is also bounded by $O(n)$.

3 Conclusions

We have presented a sequence of algorithms for an election in a complete graph where each algorithm was devised to circumvent the problems of its predecessors. This sequence concludes with algorithm C, which has time complexity $O(n)$ and message complexity $2 \cdot n \cdot \log n + O(n)$. Yet, the major contribution of the paper is the study of various methods we believe to be recurrent in a number of election algorithms. We have combined these methods to derive an algorithm better than those employing any one of the methods alone. Recently [3] ,we have applied this to derive an $O(\log n)$ time and $O(n \cdot \log n)$ messages, synchronous election algorithm for a complete network. We are now in the process of investigating the ramification of this idea to election in general networks.

References

[1] Greg N. Frederickson and Nancy A. Lynch, "The Impact of Synchronous Communication on the Problem of Electing a Leader in a Ring," (Nov. 83). Detailed Abstract.

[2] Eli Gafni and Yehuda Afek, "Election and Traversal in Unidirectional Networks," in *Proceedings of the ACM Symp. on Principles of Distributed Computing*, Vancouver BC (August 1984).

[3] Eli Gafni, Yehuda Afek, and Leonard Kleinrock, "Fast and message optimal synchronous election algorithms for a complete network," CSD-840041, UCLA, Los Angeles, CA 90024 (October 1984).

[4] Robert G. Gallager, "Finding a Leader in a Network with O(E) + O(N log N) Messages," , M.I.T. (1977). Unpublished Note.

[5] Robert G. Gallager, Pierre A. Humblet, and P. M. Spira, "A Distributed Algorithm for Minimum Weight Spanning Trees," *ACM Trans. Program. Lang. Syst.* **5**, pp.66-77 (Jan 1983).

[6] E. Korach, S. Moran, and S. Zaks, "Tight Lower and Upper Bounds for Some Distributed Algorithms for a Complete Network of Processors," in *Proceedings of the ACM Symp. on Principles of Distributed Computing*, Vancouver BC (August 1984).

[7] J. Pachl, E. Korach, and D. Rotem, "A Technique for Proving Lower Bounds for Distributed Maximum-Finding algorithms," pp. 378-382 in *Proceedings of the 14th Ann. ACM Symp. on Theory of Computing*, San Francisco CA (1982).

DISTRIBUTED ELECTION IN UNIDIRECTIONAL EULERIAN NETWORKS (SUMMARY)

ELIEZER GAFNI*
WILLARD KORFHAGE*
Department of Computer Science
University of California at Los Angeles
Los Angeles, California 90024

Consider a network containing a number of nodes, each with unique IDs. Exactly one node -- it can be any node -- must be elected as leader of the network, and all the nodes in the network must agree on the choice. This is the election problem, and in this paper we present a solution to this problem for certain types of networks.

Much previous work on the election problem has been for bidirectional networks (networks with duplex links). It is known that we can hold an election in such networks with O(N log N + E) messages simply by building a spanning tree in the network [5]. In a bidirectional network in the form of a ring, Burns [1] proved that O(N log N) is a lower bound on message complexity.

For unidirectional networks (networks with simplex links), Dolev, Klawe and Rodeh [2] and Peterson [7] have presented (for rings) election algorithms requiring O(N log N) messages. This is a lower bound on complexity, as proved by Pachl, Korach and Rotem [8] and by Fredrickson and Lynch [3]. Gafni and Afek [4] have presented an algorithm for election in an arbitrary, strongly-connected directed network using $O(NE + N^2 \log N)$ bits of communication. Here we offer an algorithm to perform an election in a unidirectional network, with the only condition being that each node in the network have an equal number of incoming and outgoing edges (i.e. an Eulerian network). Our algorithm uses O(E log N) messages; its bit complexity, $O(E \log^2 N)$, lies between that of the general algorithm and that of the ring algorithms.

THE NETWORK MODEL

We assume a simple network model. There are N nodes and E links (or edges). Links between nodes are one-way, error-free with FIFO ordering of the messages passing over a link. Every node has an equal number of incoming and outgoing links (the Eulerian condition). All nodes are programmed identically. Initially, each node knows its unique ID but does not know the IDs of its neighbors.

THE ALGORITHM

The algorithm starts with each node creating an association between its input and output edges: each output edge is associated with a unique input edge, and the node pretends there is an edge (called an internal edge) between each pair. The requirement that each node have an equal number of input and output edges guarantees that we can make this association; this requirement further guarantees that making the association will split the network into a number of circuits -- not necessarily simple circuits -- each of which will be called a cluster.

Tokens are the active entities in our algorithm; through their actions, clusters merge with each other until only one cluster is left in the network. The algorithm proper starts when each node, after making its input-output association, creates a token.

All clusters are initially tokenless, and all tokens initially control a null cluster (nothing). Tokens, using Dolev's or Peterson's election algorithm for directed rings, then compete for control of tokenless clusters by holding an election within the cluster. When a token

* This research supported by DARPA contract MDA 903-82-C-0064. The second author is also supported by an HP/AEA Fellowship.

gets control of a cluster, it claims that cluster as its own by sending a message around the cluster's Eulerian circuit (it may be seen inductively that a cluster always has an Eulerian circuit). Once this has finished, the token traverses its cluster, looking for other tokenless clusters for which it can compete and looking for other already-claimed clusters with which it can merge.

Two clusters may merge if they are adjacent (pass through the same node). Merges are executed simply by altering the input-output association (crossing two appropriate internal edges) at a node where the clusters are adjacent.

The order of merging is controlled by the clusters' IDs and levels in a method similar to Gallager's algorithm. A cluster can merge immediately with any cluster that has a higher level. A cluster cannot merge with a cluster with lower level, but must wait for the latter's level to grow to at least the level of the former. Clusters of equal level are allowed to merge, but only the cluster with the higher ID can perform the merge (otherwise, deadlocks may result). Some message traffic is required to do the merging in an orderly fashion; details appear in [6].

When a token has completely traversed its cluster and can find no other clusters with which to merge, the algorithm ends. At that point, the node that started the surviving token is elected, and all the edges in the network are contained in one cluster.

To compute the message complexity, we note that messages are used to hold elections, to control mergings and to represent tokens. The worst case number of messages for elections occurs if the input-output associations create one cluster containing all the edges. In this case, we have formed a ring containing E edges with N nodes active in the election, and the election will require O(E log N) messages.

To account for messages used in merging and to represent tokens, we show that each edge in the graph will be claimed by clusters of successively higher levels and the number of such levels is no more than log N. At a given level, the number of messages that pass over an edge is bounded by some constant. The total message complexity for mergings, therefore, is O(E log N), and the algorithm, as a whole, also requires O(E log N) messages.

REFERENCES

[1] Burns, James E. "A Formal Model for Message Passing Systems," Technical Report No. 91, Computer Science Dept., Indiana University, 1980.

[2] Dolev, Danny, Maria Klawe, and Michael Rodeh. "An O(n log n) Unidirectional Distributed Algorithm for Extrema Finding in a Circle," *Journal of Algorithms*, vol. 3, 1982, pp. 245-260.

[3] Fredrickson, Greg. N., and Nancy A. Lynch. "The Impact of Synchronous Communication on the Problem of Electing a Leader in a Ring," MIT/LCS preprint, 1983.

[4] Gafni, E. and Y. Afek. "Election and Traversal in Unidirectional Networks," Proc. 3rd Symposium on Principle of Distributed Computing, 1984, pp. 109-198.

[5] Gallager, R.G., P.A. Humblet, and P.M. Spira. "A Distributed Algorithm for Minimum-Weight Spanning Trees," *ACM Tran. on Prog. Lang. and Systems*, January 1983, pp. 66-77.

[6] Korfhage, Willard R. "Distributed Election in Unidirectional Networks, With Applications," forthcoming UCLA Master's Thesis.

[7] Peterson, Gary L. "An O(n log n) Unidirectional Algorithm for the Circular Extrema Finding Problem," *ACM Tran. on Prog. Lang. and Systems*, Oct. 1982, pp. 758-762.

[8] Pachl, J., E. Korach, and D. Rotem. "A technique for Proving Lower Bounds for Distributed Election Algorithms," *Proc. 14th Symposium on Theory of Computing*, 1982, pp. 372-382.

PROBLEMS IN BROADCASTING AND GOSSIPING (EXTENDED ABSTRACT)

DANA RICHARDS
Department of Computer Science
University of Virginia
Charlottesville, Va. 22901 U.S.A.

ARTHUR L. LIESTMAN †
Department of Computing Science
Simon Fraser University
Burnaby, B.C. V5A 1S6 Canada

ABSTRACT

Three generalizations of broadcasting and gossiping are considered: conference broadcasting in which broadcasting is done by conference calls, partial gossiping in which every gossip must learn at least k pieces of information, and set to set gossiping in which one subset of the members must learn all of the messages of another subset of the members.

INTRODUCTION

Gossiping and broadcasting are problems which have been studied extensively [3]. In an attempt to understand the structure of these problems we consider some natural generalizations in this paper. The problems are defined for a population of n members, each knowing a unique message and ignorant of the messages of the other members at the start of the process. These messages are spread by **calls**. During each call, all of the information known to the individual members of the call (callers) is learned by all of the callers. A typical call will involve two callers, but we also consider **conference calls** involving k members for $k>2$. Note that conference calls require k members in every call. No member may be in two calls simultaneously. We assume that each call can be completed in one unit of time. We use a graph (or hypergraph for conference calls) to represent those calls that are allowed. A complete graph indicates that any node (member) may call any other node. The complete graph is assumed unless otherwise stated.

Broadcasting is the problem of informing every node of the message of one node, the **originator**. It clearly requires $n-1$ calls to broadcast, however, if we allow calls to occur simultaneously the **broadcast time** can be much shorter. It is easy to see that $\lceil \log_2 n \rceil$ time is necessary and sufficient for broadcasting. **Gossiping** is the problem of informing every node of the messages of every other node. In the complete graph, with no further restrictions, $2n-4$ calls are necessary and sufficient for $n \geq 4$. If the graph of allowed calls is connected but does not contain a cycle of length 4, a C_4, then $2n-3$ calls are necessary and sufficient. We do not consider the use of simultaneous calls for gossiping.

Three generalizations are considered in this paper. We begin by looking at **minimum conference broadcast hypergraphs**. Such hypergraphs are the sparsest hypergraphs which allow every node to broadcast in optimal time. Second, we look at **partial gossiping** in which every node must become informed of any k distinct messages, including its own. When $k=n$, this is gossiping. We also consider **set to set broadcasting**. Two, possibly intersecting, sets of nodes are selected. Each node of the second set must be informed of every message of the first

† This research is supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. A-1734.

set. When the first set contains only one node and the second set contains all of the nodes, this is broadcasting. When both sets contain all of the nodes, we have gossiping. For this problem and for partial gossiping we use only two-person calls and are concerned with the total number of calls.

MINIMUM CONFERENCE BROADCAST HYPERGRAPHS

Let $t_k(n)$ be the time required to broadcast from one of n nodes using the complete k-uniform hypergraph on n vertices as the hypergraph of allowed conference calls. Since it is easy to see that $t_2(n)=\lceil \log_2 n \rceil$ it is surprising that $t_k(n)$ is not always $\lceil \log_k n \rceil$. The following lemma indicates the effect of requiring every call to include k nodes.

Lemma 1: Let $n = k^i + p(k-1) + q$ with $1 \leq p < k^i$, and $r = k^i - p$.

$$t_k(n) = \begin{cases} \lceil \log_k n \rceil & \text{for } \max\{0, k-r\} \leq q < k \\ \lceil \log_k n \rceil + 1 & \text{for } 1 \leq q < k-r \end{cases}$$

Proof: Since at most k^j nodes can be informed after j time units, $t_k(n) \geq \lceil \log_k n \rceil = i+1$. It is easy to provide tight constructive upper bounds for the two cases. In both cases, begin by informing k^i nodes, and then have p of these each call $k-1$ other nodes. Note that r is the number of the k^i still available. In the first case, $q+r \geq k$ so the remaining q can be covered by one edge. In the second case, we use an additional time unit to cover these q nodes. To complete the proof we need to establish the lower bound for the second case.

We proceed by induction on n ; the basis is trivial. Assume n is of the form with $1 \leq q \leq k-r$ and that $t_k(n)=i+1$. We claim that we must inform k^i nodes in the first i time units. Otherwise, by the hypothesis, after i time units we can inform at most k^i-k+1 nodes, from which we can reach only $k^{i+1}-k^2+k$ nodes. However, the least such n is $k^{i+1}-k^2+3k-1$. Hence, during the $(i+1)st$ time unit $p(k-1)+q$ nodes must be reached, requiring $p+1$ calls covering $(p+1)k$ nodes. However, $n=pk+q+r$ and $q+r < k$ so $n < (p+1)k$ giving a contradiction. □

A **minimal conference broadcast hypergraph,** G, is a hypergraph of allowed calls which permits every node to be the originator of a $t_k(n)$ time broadcast, but the removal of any edge from G produces a hypergraph which does not permit this. $B_k(n)$ is the least number of edges in any such hypergraph. A **minimum conference broadcast hypergraph, (mcbh)** is a minimal conference broadcast hypergraph with $B_k(n)$ edges.

For $k=2$ only a few values of $B_2(n)$ are known [1], i.e. for $1 \leq n \leq 16$ and for $n=2^i$. The proofs for even these small values of n are quite tedious; the tedium increases with k. As a consequence we have only a few proofs of tight lower bounds on $B_k(n)$, for $k=3$ and $k=4$. We have, however, produced constructive upper bounds which we conjecture are the best possible. First we prove a result analogous to the known results for $B_2(2^i)$.

Lemma 2: $B_k(k^i) = ik^{i-1}$.

Proof: Since $t_k(k^i)=i$, a simple inductive argument shows that the originator must be involved in i separate calls. Since every node can be an originator we get a lower bound by summing the

degrees of the vertices. So $B_k(k^i) \geq \frac{1}{k}(ik^i) = ik^{i-1}$.

We give a simple inductive construction for the upper bound. Build k mcbh's on k^{i-1} nodes each and add k^{i-1} disjoint edges, each edge intersecting each of the k mcbh's. This uses $k((i-1)k^{i-2})+k^{i-1} = ik^{i-1}$ edges. □

For k=3,4 our upper bounds are summarized in Table 1. Some of the corresponding hypergraphs are shown in Figures 1 and 2. Note that when the time "increases" we often find a drop in $B_k(n)$ reflecting the increase in flexibility. For example, $B_3(7) = 5$ while $B_3(8) = 4$. What is surprising is that even in ranges where $t_k(n)$ is constant $B_k(n)$ may decrease with an increase in n. This seems to occur for k=3 at n=15. This anomaly is due to our requiring k members per call.

In Table 1 asterisks indicate those values for which we can show corresponding lower bounds. We will omit the proofs here and will present them in a forthcoming report. The key in such a proof is to study the pattern of edges necessary for a single originator. For example, for n=12 and k=3 if the originator is in the leftmost vertical edge the edges it requires are isomorphic to the hypergraph formed by deleting the 2 rightmost vertical edges. A single edge cannot be added to this hypergraph to form a minimal conference broadcast hypergraph. For larger n the number of cases tends to increase.

The hypergraphs in the figures require little annotation, though in some cases it may be hard to discern each of the n broadcast schemes. For small values of n the graphs are trivial and are not shown. For certain ranges of n there are schemas which were used to find these graphs and it is clear that for larger k the same schemas can be used , e.g. $B_3(11)$ and $B_4(19)$. When $t_k(n)$=2 there is little flexibility.

When $t_k(n)$=3 we have found the following schema to be fecund. If $n=pk$, partition the nodes into p edges. Add additional edges so that each of the p edges is connected to all of the others by disjoint edges. To broadcast, begin with one of the p edges, then broadcast to the other $p-1$ edges in the second time unit, and finally make the other $p-1$ calls. If $n+pk+q$, $0<q<k$, then handle the q singletons in an ad hoc manner. This schema can only work if $n \leq k+(k(k-1))k = k^3-k^2+k$. The culmination of this is seen in the following result.

Lemma 3: If there exists a symmetric balanced incomplete block design with parameters k and λ , λ=1 then $B_k(k^3-k^2+k) \leq 2(k^2-k+1)$.

Proof: Recall that a symmetric balanced incomplete block design with parameters v, k, and λ, λ=1, has v objects in v blocks of k objects each, and each object appears a total of k times. Further, no pair of objects occurs in two blocks. It is known ([2], p.104) that $v=k(k-1)+1$.

We construct a hypergraph with $n=vk$ nodes by partitioning the nodes into v edges, and identifying these edges with the objects. Add v disjoint new edges corresponding to each block. Note that the properties of the design ensure that exactly k new edges are incident with an old edge and they span the other old edges. Note that $t_k(n)$=3, so we can use the aforementioned schema. □

There are values of k for which there are no symmetric balanced incomplete block designs, leaving the corresponding cases open. The designs exist for k=3,4,5,6,7,8,9,10,12 and are known not to exist for k=7,15 ([2], Appendix A). Note

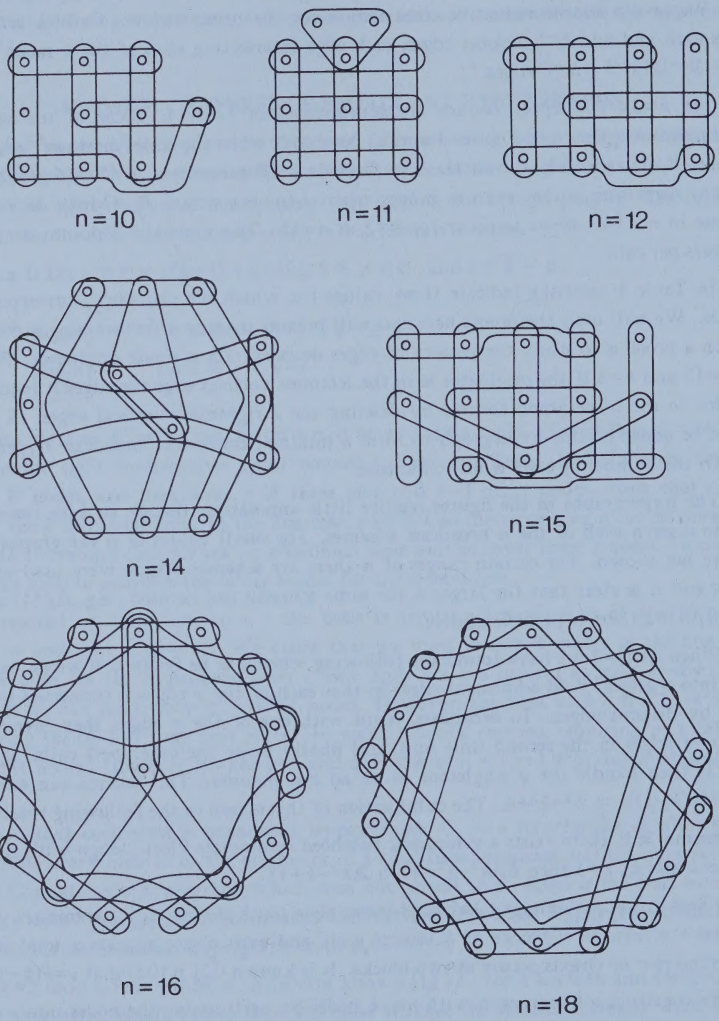


Figure 1: minimal conference broadcast hypergraphs for $k=3$

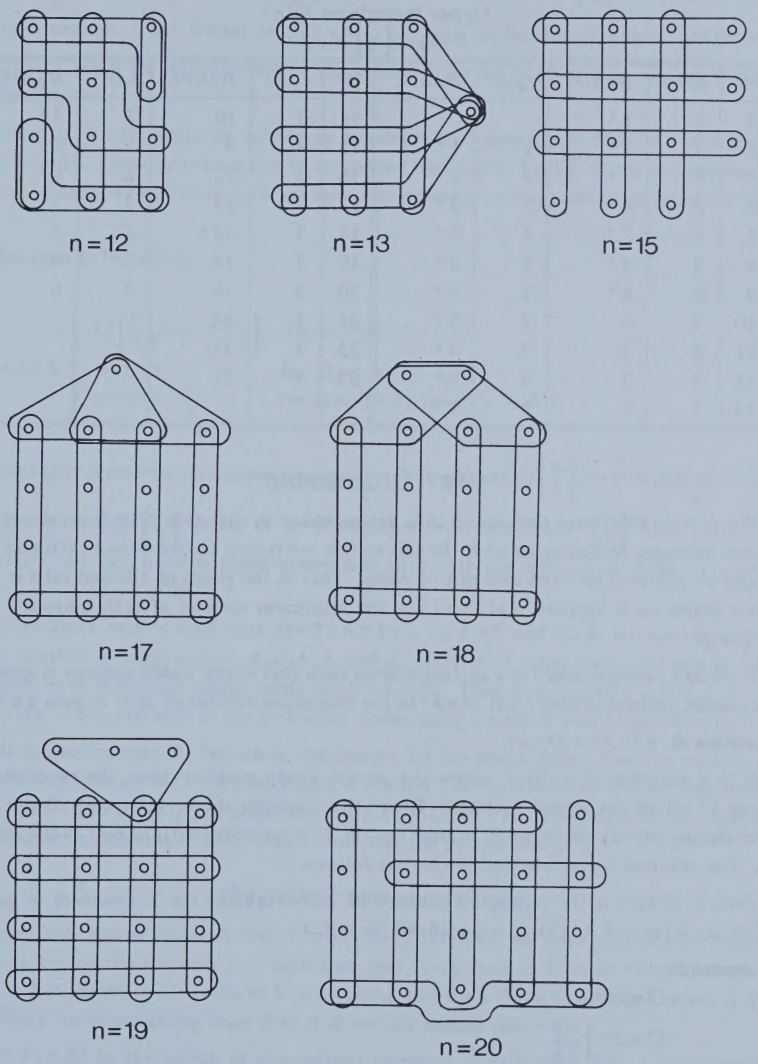


Figure 2: minimal conference broadcast hypergraphs for $k=4$

that in the above calling scheme the original k nodes are idle during the third time unit, permitting them to call the q others, if $q \neq 0$. This insight gives hypergraphs for $k=3$ and $n=20, 22, 23$. We also note that $B_4(24) \leqslant 9$ using a hypergraph similar to that for $B_4(20)$.

Upper Bounds on $B_k(n)$
(* indicates best possible)

n	$t_3(n)$	$B_3(n) \leqslant$	$t_4(n)$	$B_4(n) \leqslant$	n	$t_3(n)$	$B_3(n) \leqslant$	$t_4(n)$	$B_4(n) \leqslant$
3	1	1 *			14	3	10	3	5 *
4	2	2 *	1	1 *	15	3	9	3	6 *
5	2	2 *	2	2 *	16	3	12	2	8 *
6	2	4 *	2	2 *	17	3	13	3	7
7	2	5 *	2	2 *	18	3	12 *	3	7
8	3	4 *	2	4 *	19	3	14	3	9
9	2	6 *	2	5 *	20	3	18	3	8
10	3	6	2	5 *	21	3	14		
11	3	7	3	4 *	22	3	21		
12	3	7	2	6 *	23	3	21		
13	3	8	2	7 *					

PARTIAL GOSSIPING

We wish to determine how many calls are necessary so that each node is informed of any k distinct messages, including its own. In this section we return to two-person calls and assume that calls are allowed between any pair of nodes. That is, the graph of allowed calls is K_n, the complete graph on n vertices. Let $C(n,k)$ be the minimum number of calls necessary to partially gossip.

A related problem asks for a calling scheme such that every node's message is spread to k distinct nodes, including itself. Let $D(n,k)$ be the minimum number of calls in such a scheme.

Lemma 4: $C(n,k) = D(n,k)$.

Proof: If a sequence of calls S solves the partial gossip problem then the reversal of the sequence S^R solves the second problem. To see this, consider the $k-1$ messages that a node x receives during partial gossiping. It is clear that in S^R x 's message will be sent to the same $k-1$ nodes. The converse is also true and the lemma follows. □

Clearly $C(k,k)$ is the minimum number of calls required for k members to gossip, so $C(k,k) = 2k-4$ for $k \geqslant 4$. $C(n,k)$ is undefined for $n < k$.

Lemma 5:

$$C(n,1)=0.$$

$$C(n,2)=\left\lceil \frac{n}{2} \right\rceil.$$

$$C(n,3)=\left\lceil \left\lfloor \frac{n}{2} \right\rfloor /2 \right\rceil + \left\lceil \frac{n}{2} \right\rceil.$$

Proof: $C(n,1) = 0$ is trivial. The value for $C(n,2)$ follows from the observation that each node

must be involved in one call. From the above remarks about $C(k,k)$ we know that $C(3,3)=3$ which coincides with the value given in the Lemma. For $C(n,3)$ for $n>3$, we note that the final call for each node must involve at least one node which knows at least two messages. The minimum number of final calls is $\lceil \frac{n}{2} \rceil$ in which case we require $\lfloor \frac{n}{2} \rfloor$ nodes knowing more than one message. That would require $C(\lfloor \frac{n}{2} \rfloor,2)$ calls in the initial phase. Simple algebra shows that if fewer (or more) calls were made during the initial phase then no fewer total calls would result. □

Consider $C(8,4)$. Begin by informing 4 nodes with 3 messages in $C(4,3)=3$ calls and making 4 final calls. Thus the process can be completed using only 7 calls. If we had started with a matching, 8 calls would have been needed. Generalizing this approach we get the following result.

Theorem 6: For $k \geq 4$,

$$C(n,k) \leq \begin{cases} C(\lfloor \frac{n}{2} \rfloor,k-1)+\lceil \frac{n}{2} \rceil & \text{for } n \geq 2k-2 \\ n+k-5 & \text{for } 2k-2 > n > k+2 \\ n+k-4 & \text{for } \min\{k+2, 2k-2\} > n \geq k \end{cases}$$

Proof: We give a constructive upper bound. If $n \geq 2k-2$ then $C(\lfloor \frac{n}{2} \rfloor,k-1)$ is defined so we can begin by informing $\lfloor \frac{n}{2} \rfloor$ nodes of $k-1$ messages and then make $\lceil \frac{n}{2} \rceil$ final calls. When $2k-2>n \geq k$ we can have k nodes gossip and then call the other $n-k$ nodes in a total of $C(k,k)+n-k = n+k-4$ calls.

When there exist n such that $2k-2>n>k+2$, i.e. $k>5$ and $n>8$, we use the following scheme. Identify 8 nodes $a, b, c, d, \hat{a}, \hat{b}, \hat{c}$, and $\hat{d}$. Have $k-5$ other nodes call any of a,b,c, or d and then have these four gossip, using $(k-5)+4$ calls. Then a, b, c, and d call $\hat{a}, \hat{b}, \hat{c}$, and $\hat{d}$, respectively. Then call each of the $n-8$ other nodes, using a total of $k-1+4+n-8$ calls. □

We conjecture that the bounds in this lemma are the best possible. Careful reading of the proof indicates that we use a cycle of length 4 when $n<2k-2$. However it is not required since we need only those edges in the graph in Figure 3 for $C(8,6)=10$. The labels on the edges in the figure indicate an ordering of calls which completes the partial gossip.

SET TO SET BROADCASTING

Given two sets of nodes, A and B, which may intersect, we wish to broadcast from A to B. That is, we want a sequence S of calls such that every node in B learns every message from the set A. Let the set of all nodes be V with $|A|=k$ and $|B|=l$. Note that, as in Lemma 4, S^R is a valid scheme for broadcasting from B to A so we can assume that $k \leq l$.

Let $F(A,B)$ be the length of the shortest sequence of calls such that A broadcasts to B, assuming that all calls are possible. If the set of allowed calls are constrained to the edges of a given graph G, we define $F(A,B,G)$ analogously. The following result summarizes our findings.

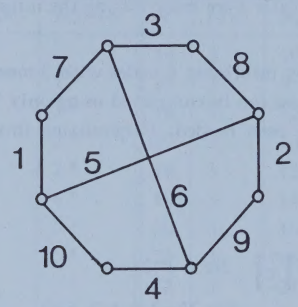


Figure 3: calls for $n=8$, $k=6$

Lemma 7: Let $i=|A \cap B|$.

$$F(A,B) = \begin{cases} k+l-i-1 & \text{for } 0 \leqslant i \leqslant 2 \\ k+l-3 & \text{for } i=3 \\ F(A,B) \leqslant k+l-4 & \text{for } i \geqslant 4. \end{cases}$$

Proof: We first establish the upper bounds. For $i=0$ have each node in A call some node $b \in B$ and then have b broadcast the messages within B. For $i=1$ the same scheme will work if $b \in (A \cap B)$. For $i \geqslant 2$, let $b \in (A \cap B)$ and have all nodes in $A \setminus (A \cap B)$ call b. The i nodes in the intersection can gossip among themselves and then call the nodes in $B \setminus (A \cap B)$.

The set of edges corresponding to the calls used in the set to set broadcast must form a connected graph over the vertices in $A \cup B$. This requires $k+l-i-1$ edges (or calls). Thus the upper bounds for $0 \leqslant i \leqslant 2$ are the best possible.

Consider a tree of $k+l-4$ edges connecting the nodes in the case $i=3$. If each edge represents a single call, it is not possible for all three members of the intersection to learn each other's message. In this case there would be a pair of nodes $a, b \in (A \cap B)$ separated by at least two edges in the tree. If a learned b's information in the scheme then the calls between a and b would occur in temporal order from b to a and b could not learn a's information. Thus at least $k+l-3$ calls are needed for $i=3$. □

We conjecture that for $i \geqslant 4$ our result is the best possible.

For general graphs G broadcasting may take much longer. We would like to characterize those G such that $F(A,B) = F(A,B,G)$ for all subsets A,B of the vertices V. For the rest of this section we will restrict ourselves to the case where A and B are disjoint, i.e. $i=0$. Therefore we know that $F(A,B)=k+l-1$.

Let $\boldsymbol{H}(k,l)$ be the set of all graphs G on n vertices such that $F(A,B,G)=F(A,B)$ for all disjoint A and B. Let $h(k,l)$ be the minimum number of edges in a graph from $\boldsymbol{H}(k,l)$. For $k+l \leq 4$ we have strong characterizations.

Lemma 8:

$$h(1,1) = \binom{n}{2}.$$

$$h(1,2) = \binom{n}{2} - \left\lfloor \frac{n}{2} \right\rfloor.$$

$$h(1,3) = h(2,2) = \begin{cases} \binom{n}{2} - n & \text{for } n > 4 \\ 3 & \text{for } n = 4 \end{cases}$$

Proof: For any graph G in $\boldsymbol{H}(k,l)$, consider the dual G'. If G is in $\boldsymbol{H}(1,1)$ and G' contains any edge (x,y), then x cannot broadcast to y. If G is in $\boldsymbol{H}(1,2)$ and G' contains a vertex x adjacent to two distinct vertices y and z, then x cannot broadcast to y and z in G using only two calls. Hence, G' can contain no vertices of degree >1 and can have at most $\left\lfloor \frac{n}{2} \right\rfloor$ edges.

G is in $\boldsymbol{H}(1,3)$ or $\boldsymbol{H}(2,2)$ if and only if the induced subgraph of any 4 vertices of G' is not $K_{1,3}$ or C_4. This can be verified by case analysis. Note that if G' contains a $K_{1,3}$ or a C_4 the corresponding subgraph of G is disconnected. Since no vertex in G' has degree >2, G' is at most a collection of disjoint cycles, except when $n=4$. From these observations the lemma follows. □

Notice that for each choice of A and B the $k+l$ node induced subgraph of a G in $\boldsymbol{H}(k,l)$ must contain some spanning tree T which is used for broadcasting. Consider such a tree T with k nodes labelled a and l nodes labelled b. Consider the following two reductions. Remove all leaves labelled the same as their adjacent node. Remove all but one of the remaining leaves at each internal node. Repeat this reduction if necessary to form T''.

Lemma 9: The a's can broadcast to the b's in T if and only if they can in T''.

Proof: It is easy to verify this as we remove each node separately. □

By exhaustion we find that the only reduced trees for <6 nodes are the paths of lengths 2, 3, 4, and 5. Of these only the path of length 5, $P_5=(v_1, v_2, v_3, v_4, v_5)$, can be labelled so that broadcasting cannot be done in the minimum time. The difficulty occurs when both extreme edges, (v_1, v_2) and (v_4, v_5), have distinct labels on their endpoints, e.g. (a, b, a, b, a). In these cases v_3 becomes a bottleneck.

A graph G is ***P_5–like*** if it satisfies the following condition. For some choice of A and B every spanning tree of the $k+l$ labelled vertices contains two separated edges, both with their endpoints in A and B. Two edges are **separated** if they are disjoint and not connected by an edge.

Theorem 10: G is in $\boldsymbol{H}(k,l)$ if and only if it is not *P_5–like* and each $k+l$ node induced subgraph is connected.

Proof: (if) Such a graph has, for any choice of A and B, a corresponding reduced spanning tree that is a path of length 4 or less and by the previous lemma broadcasting is possible in G.

(only if) For every A and B the $k+l$ node induced subgraph of G must be connected by a spanning tree T used for broadcasting. If T has diameter >4, it is P_5–*like* and broadcasting is not possible in G. □

Perhaps there is a more succinct characterization of $\boldsymbol{H}(k,l)$ in terms of spanning trees or diameter. However, the cycles of length 5 and 6 complicate the situation. The former is in $\boldsymbol{H}(2,3)$ while the latter is not in $\boldsymbol{H}(3,3)$.

REFERENCES

1. Farley, A.M., S.T. Hedetniemi, S. Mitchell and A. Proskurowski, "Minimum Broadcast Graphs", Discrete Mathematics, Vol. 25, 1979, pp. 189-193.

2. Hall, M., Jr., Combinatorial Theory, Wiley, New York, 1967.

3. Hedetniemi, S.T. and S.M. Hedetniemi, "A Survey of Gossiping and Broadcasting in Communication Networks", Technical Report CIS-TR-81-5, University of Oregon, 1981.

FH-MFSK SYSTEMS.

G.E.ATKIN I.F.BLAKE
Electrical Engineering Department
University of Waterloo
Waterloo, Ontario, N2L-3G1.

ABSTRACT

A new technique that allows the communicator the use of several frequencies for the transmission of information in a Frequency Hopped M-ary Frequency Shift Keying system is analysed. The frequencies used by the communicator are selected from the blocks of a Steiner Triple System S(t,k,v) and are distributed over the available bandwidth in different manners. The performance of the system is evaluated considering different jammer strategies. It is shown that some gains are obtained with respect to a conventional Frequency Hopped-MFSK system.

1. INTRODUCTION

Frequency Hopped Spread Spectrum (FH-SS) Systems that use M-ary waveforms with diversity, or coding have gained increased popularity in the design of anti-jam communication systems. The primary reason for the use of orthogonal sets of signals with coding, is the robustness in performance that this technique achieves under different types of unintentional or hostile interference.

Several forms of interference have been extensively studied using the above systems, allowing the jammer to distribute its power over the available bandwidth in order to optimize its performance, and restricting the communicator to use only one tone per M-ary band of frequency [1]. In this paper a different technique is analysed, and some gains are obtained with respect to conventional FH-MFSK systems.

The proposed technique is based on the use of several tones for the transmission of the information. The communicator chooses to spread the frequencies that he has available over the bandwidth W, according to different strategies in order to use more effectively its power. The first scheme to be considered is the use of simple multiple frequencies, i.e., instead of using one tone per slot in the transmission of the information, several frequencies are distributed in different slots, allocating one of the above frequencies per slot, as shown in figure 1.

2. MULTIPLE FREQUENCY TRANSMISSION

The probability for the communicator to be confused by the jammer is obtained using L_f tones distributed over L_f slots. The results are compared with a conventional FH-MFSK system.

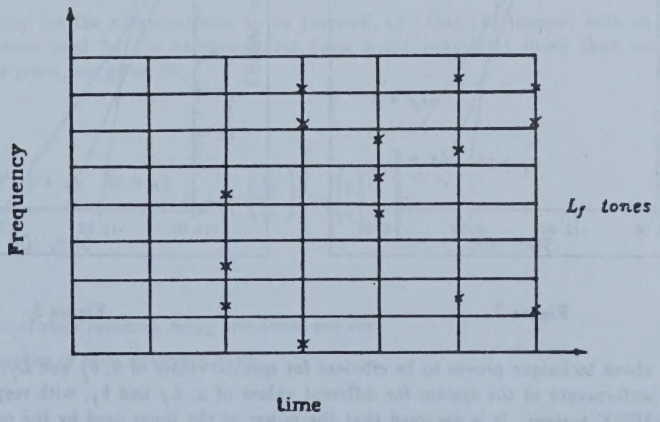


Figure 1.

The probability for the communicator being confused,i.e., that the jammer successfully interferes in the L_f slots in frequency is given by:

$$P_{rm} = \left\{ \frac{\binom{n - L_f}{b_f - L_f}}{\binom{n}{b_f}} \right\} \tag{1}$$

where

$b_f \geq L_f$ = number of tones used by the jammer

P_{rm} = Probability for the communicator being confused

n = number of slots available in bandwidth W

For a conventional FH-MFSK system, the probability of being confused is given by:

$$P_r = \left\{ \frac{b}{n_r} \right\} = \left\{ \frac{1}{\gamma n_r} \right\} \tag{2}$$

where

b = number of tones used by the jammer

n_r = number of slots available in bandwdth W

γ = signal to noise ratio

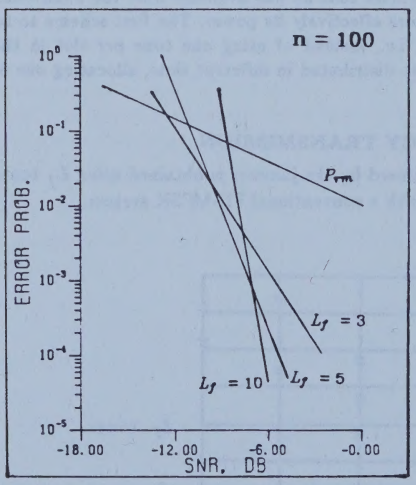


Figure 2.

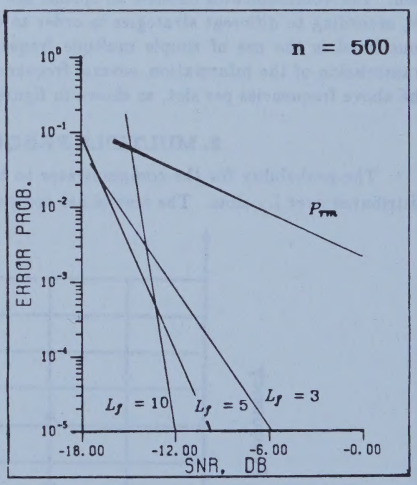


Figure 3.

The above technique proves to be efficient for specific values of n, b_f and L_f. Figures 2 and 3 show the performance of the system for different values of n, L_f and b_f, with respect to a conventional FH-MFSK system. It is assumed that the power of the tones used by the communicator and the jammer is the same.

The next scheme to be considered makes use of Steiner Triple Systems to select the different tones in a more sophisticated manner than diversity.

3. FH-STEINER-MFSK SYSTEM

Assume that M v-tuples are available, each of weight k, and having the property that the number of other v-tuples that intersect a given v-tuple in "i" places is independent of the v-tuple that has been chosen. To obtain the set of M v-tuples needed, a Steiner Triple System is used [2].

A Steiner System S(t,k,v), is an arrangement of v distinct elements into b blocks, such that each block contains k elements, and any subset of t distinct elements is contained in exactly one block [2]. Steiner systems with parameters t = 2, k=3, and v=13, 15 will be considered. The number of blocks b, and the number of blocks intersecting a given block in "i" points x_i, i=1,2,3,.....,(k-1), are given in equations (3),(4) and (5).

$$b = \frac{v(v-1)}{6} = \textit{number of blocks, each with k elements} \tag{3}$$

$$x_1 = \frac{3(v-3)}{2} = \textit{number of blocks with one intersection} \tag{4}$$

$$x_0 = b-1-x_1 = \textit{number of blocks without intersection} \tag{5}$$

Depending on the K bits to be transmitted, one of the $b \geq M$ blocks is selected. Once the Steiner block has been chosen, different strategies for the communicator and jammer will be considered. In particular, three strategies will be analysed for a Steiner System with block length k=3 (other values of k may be considered, but the complexity of the analysis increases).

First, given the Steiner block previously selected, the communicator transmits k of the v orthogonal frequencies available, all in the same slot, and the jammer chooses to interfere using (k-1) tones per slot in order to possibly form another block of the Steiner systems in use. This can be achieved using one of the tones that the communicator is transmitting and the two (k=3) tones of the jammer, that will allow to form one of the blocks that have one intersection with the one that has been used by the communicator. In the second case, the communicator uses the same strategy as in the first case, but the jammer uses now k tones per slot. For the third case, the communicator uses k different slots, distributing one tone per slot, and the jammer uses one tone per slot to interfere the transmission. In all the cases, it is assumed that the jammer and the communicator use the same tone power.

The probability for the communicator to be jammed, i.e., that the jammer with its tones together with the ones used by the communicator form a set containing more than one possible transmitted sets of tones, are given by:

First case.

$$P_{r1} = \left\{\frac{b_2}{n}\right\}\left\{x_1\binom{v}{2}^{-1}\right\} \tag{6}$$

where

b_2 = number of slots jammed, using two tones per slot

n = total number of slots available in W

Second case.

$$P_{r2} = \left\{\frac{b_3}{n}\right\}\left\{(x_0+3x_1+\alpha)\binom{v}{3}^{-1}\right\} \tag{7}$$

where

b_3 = number of slots jammed, using three tones per slot
α = parameter that depends on the Steiner System used

Third case.

$$P_{r3} = \left\{x_1\binom{v}{2}^{-1}\right\}\left\{k\frac{\binom{n-k}{b_1-k+1}}{\binom{n}{b_1}}\right\} + \left\{(x_0+3x_1+\alpha)\binom{v}{3}^{-1}\right\}\left\{\frac{\binom{n-k}{b_1-k}}{\binom{n}{b_1}}\right\} \tag{8}$$

where

b_1 = number of slots jammed, one tone per slot

Figures 4 and Figure 5 show the improvement obtained for the different cases with respect to conventional FH-MFSK, as a function of the signal to noise ratio (S/J=γ). For the above case, i.e., FH-MFSK with one tone, the probability of being jammed is given by:

$$P_{rr} = \left\{\frac{\beta}{n_r}\right\} \tag{9}$$

where

$n_r = \left\{\frac{nv}{2^K}\right\}$ = total number of slots in W

β = total number of tones used by the jammer

Equations (6), (7), (8) and (9) can be expressed as a function of the signal to noise ratio (γ). For the first case ,it is obtained:

$$P_{r1} = \left\{x_1\binom{v}{2}^{-1}\right\}\left\{\frac{k}{2\gamma n}\right\} \tag{10}$$

where

$\gamma = \frac{S}{J} = \left\{\frac{kS_t}{2b_2J_t}\right\} = \left\{\frac{k}{2b_2}\right\}$ = signal to noise ratio

S_t = power of the tone used by the communicator

J_t = power of the tone used by the jammer ($J_t = S_t$)

For the second case:

$$P_{r2} = (x_0+3x_1+\alpha)\binom{v}{3}^{-1}\left\{\frac{1}{\gamma n}\right\} \tag{11}$$

where

$$\gamma = \left\{\frac{k}{3b_3}\right\} = \text{signal to noise ratio}$$

For the third case:

$$P_{r3} = \left[\frac{9x_1(3-\gamma)(n\gamma-3)}{n(n-1)(n-2)\gamma^3}\right]\binom{v}{2}^{-1} + 3(x_0+3x_1+\alpha)\binom{v}{3}^{-1}\left[\frac{(3-\gamma)(3-2\gamma)}{n(n-1)(n-2)\gamma^3}\right] \tag{12}$$

where

$$\gamma = \left\{\frac{k}{b_1}\right\} = \text{signal to noise ratio}$$

Finally, for conventional FH-MFSK:

$$P_{rr} = \left\{\frac{\beta}{n_r}\right\} = \left\{\frac{2^K}{v\ \gamma\ n}\right\} \tag{13}$$

where

$$\gamma = \left\{\frac{1}{\beta}\right\} = \text{signal to noise ratio}$$

4. EXAMPLE

To illustrate the ideas of the last section, two examples are presented using a Steiner System with parameters t=2, k=3 and v=13, 15. A comparison with conventional FH-MFSK is shown in figures 4 and 5. It is assumed that the available bandwidth W is fixed, and the number of slots n, varies according to the system that is being used, i.e., when a Steiner system is used, $n = \frac{W}{v}$, and in a conventional FH-MFSK system $n_r = \frac{W}{2^K}$ ($2^K \leq b$). Only the case with v=15 will be presented with some details, the case for v=13 is obtained using the Steiner System given in [3].

A Steiner system on 15 elements can be constructed using different techniques. A particular realization has been obtained using an expurgated Hamming code with parameters $n = 2^m-1 = 15$ and $K = 2^m-1-m = 11$. The $b = 35$ tuples on $V = \{ 1, 2, 3,.....,15 \}$ are:

$$\begin{bmatrix} (1,14,15) & (1,7,12) & (6,7,15) & (10,11,15) & (4,6,9) \\ (2,13,15) & (2,4,14) & (6,10,13) & (1,2,3) & (4,7,8) \\ (3,13,14) & (2,9,12) & (8,9,15) & (1,5,6) & (4,5,10) \\ (5,12,15) & (3,4,15) & (8,10,14) & (2,5,8) & (2,7,10) \\ (6,12,14) & (3,10,12) & (4,11,12) & (3,6,8) & (1,8,11) \\ (8,12,14) & (5,7,14) & (7,11,13) & (1,9,10) & (2,6,11) \\ (1,4,13) & (5,9,13) & (9,11,14) & (3,7,9) & (3,5,11) \end{bmatrix} \quad (14)$$

Using the above Steiner system, it is observed (figure 4) that some gains can be obtained relative to a conventional FH-MFSK system.

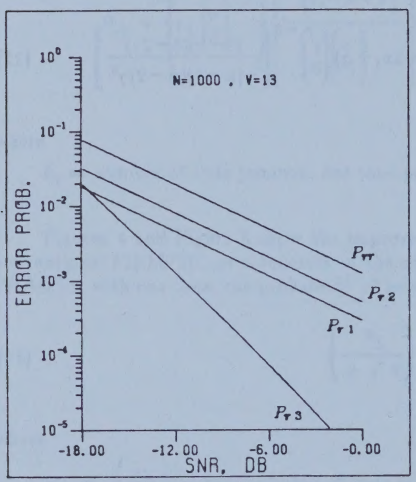


Figure 4.

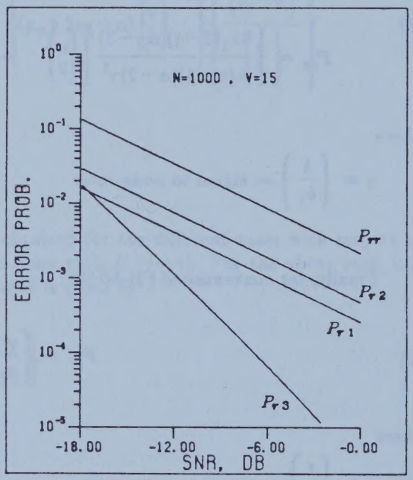


Figure 5.

5. COMMENTS

The technique described can be easily extended to a system using diversity or coding in time. The use of a convolutional encoder with Viterbi decoding has been shown to improve the performance of the system [4].

The ideas of this work can be extended to a Multiple Access System where, for a given bandwidth expansion, some gains are obtained with respect to the system proposed by Viterbi [5].

Research in the above aspects of the problem and in the evaluation of other strategies and types of codes using the proposed technique is being pursued. In addition, the performance of the system when white Gaussian noise is not negligible remains to be evaluated.

APPENDIX

First case.

Equation (6) is comprised of two terms, the first one is the probability for the jammer to hit the slot used by the communicator with two tones, given that he is interfering b_2 of the total number of slots. The second term represents the probability that given that the slot used by the communicator was hit with two tones, the communicator will be confused, i. e., the jammer will be able to form another block of the Steiner system, different to the one that the communicator is using. This is pos-

sible because the jammer can form the interfering block using one of the tones used by the communicator and his two tones, taking advantage of the fact that there are x_1 blocks in the Steiner system that have one intersection with the block used by the communicator.

$$P_{r1} = \left\{\frac{b_2}{n}\right\}\left\{x_1\binom{v}{2}^{-1}\right\} \tag{6}$$

Second case.

The second term of equation (7) is obtained as follows:

P_{r2} = Prob.(Communicator is confused / 0 overlap) Prob. (0 overlap) + Prob.(Communicator is confused / 1 overlap)

$$Prob.(0\ overlap) = \left\{\frac{\binom{v-3}{3}}{\binom{v}{3}}\right\} \tag{A1}$$

$$Prob.(1\ overlap) = \left\{3\frac{\binom{v-3}{2}}{\binom{v}{3}}\right\} \tag{A2}$$

$$Prob.(Comm.\ is\ confused/0\ overlap) = \left\{\frac{\frac{3}{2}(v-2)}{\binom{v-3}{2}}\right\} \tag{A3}$$

$$Prob.(Comm.\ is\ confused/1\ overlap) = \left\{\frac{x_0}{\binom{v-3}{3}} + \frac{\alpha_1}{\binom{v-3}{3}}\right\} \tag{A4}$$

where

$\alpha = \alpha_1 + x_0$ = parameter depending on the Steiner system used

Therefore, with (A1), (A2), (A3) and (A4), it is possible to calculate P_{r2}, i. e., the probability that the communicator is confused given that the jammer is using three tones per slot.

$$P_{r2} = \left\{\frac{b_3}{n}\right\}\left\{(x_0+3x_1+\alpha)\binom{v}{3}^{-1}\right\} \tag{7}$$

The first term in equation (7) is the probability that the jammer hits with three tones the slot that the communicator is using, given that the jammer is interfering b_3 of the total number of slots, n.

Parameter α.

In order to obtain the parameter α, the Steiner system on V = { 1, 2, 3,.....,15 } described before will be used.

The parameter $\alpha = \alpha_1 + x_0$, accounts for the probability of being confused given that there is no overlap between the tones selected by the jammer and the communicator; x_0 gives the probability of forming another block of the Steiner system without using any of the tones of the communicator, and α_1, on the other side, uses one element (tone) of the communicator's block to form another block of the system. It is known that the number of blocks that have one intersection with a given block that has been chosen by the communicator is x_1. Therefore, without lost of generality it is assumed that the block (1, 2, 3) was selected by the communicator. The set X_1, of blocks that have intersection with the communicator's block is given by:

$$X_1 = \left\{ \begin{matrix} (1,14,15) & (1,4,13) & (1,7,12) & (1,5,6) & (1,9,10) & (1,8,11) \\ (2,13,15) & (2,4,14) & (2,9,12) & (2,5,8) & (2,7,10) & (2,6,11) \\ (3,13,14) & (3,4,15) & (3,10,12) & (3,6,8) & (3,7,9) & (3,5,11) \end{matrix} \right\}$$

It is observed that there are $\frac{(v-3)}{2}$ blocks that have the first element of the communicator's block in common, $\frac{(v-3)}{2}$ with the second element and $\frac{(v-3)}{2}$ with the third element. With $(v-3)$ elements, it is possible to form a total of $\binom{v-3}{3}$ different blocks, and $\binom{v-3}{2}$ different pairs, each of these pairs is repeated $(v-5)$ times in the total number of blocks. Therefore, if the pairs of the first row of X_1 are considered, following a counting argument, there will be $\frac{(v-5)(v-3)}{2}$ blocks that will contain these pairs. Now, considering the second row of X_1, there will be $\frac{(v-7)(v-3)}{2}$ blocks left containing these pairs, and for the third row, there will be $\frac{(v-9)(v-3)}{2}$ blocks left containing the pairs corresponding to this row. Therefore, the parameter α_1 will be the sum of these blocks, i.e., $\alpha_1 = \left\{ \frac{(v-3)}{2}(3v-21) \right\} = \left\{ \frac{x_1}{3}(3v-21) \right\}$.

For the example presented in section 4, the parameter α takes the value $\alpha = \alpha_1 + x_0 = 144 + 16 = 160$.

The parameters for the Steiner system S(t, k, v) are given by:

$$b = \frac{v(v-1)}{6} = \frac{15(15-1)}{6} = 35 \qquad (3)$$

$$x_1 = \frac{3(v-3)}{2} = 18 \qquad (4)$$

$$x_0 = b - 1 - x_1 = 16 \qquad (5)$$

Third case.

Equation (8) is comprised of two terms, the first one represents the probability for the communicator to be confused, given that the jammer hits (k-1) of the slots. The second term represent the probability that the communicator is confused given that the jammer hits the k slots used by the communicator.

$$P_{r3} = \left\{ x_1 \binom{v}{2}^{-1} \right\} \left\{ k \frac{\binom{n-k}{b_1-k-1}}{\binom{n}{b_1}} \right\} + \left\{ (x_0+3x_1+\alpha) \binom{v}{3}^{-1} \right\} \left\{ \frac{\binom{n-k}{b_1-k}}{\binom{n}{b_1}} \right\} \qquad (8)$$

REFERENCES

[1] S. W. Houston, " Modulation Techniques for Communication, Part I: Tone and Noise Jamming Performance of Spread Spectrum M-ary FSK and 2, 4-ary DPSK Waveforms ", Proc. IEEE NAECON, pp. 51-58, 1975.

[2] I. F. Blake, " Performance of Nonorthogonal Signaling on an Asynchronous Multiple Access On-Off Channel ", IEEE Transactions on Communications, Vol. 1, pp. 293-298, January 1982.

[3] M. Hall, " Combinatorial Theory ". Waltham, MA: Blaisdell, 1967.

[4] G. E. Atkin, I. F. Blake, " FH-MFSK Systems using Multiple Tones ", Internal Report, University of Waterloo (to be published).

[5] A. J. Viterbi, " A Processing Satellite Transponder For Multiple Access By Low-Rate Mobile Users ", Proc. Fourth Int. Conf. on Digital Satellite Communications, Montreal, Canada, October 1978.

ERROR PERFORMANCE ANALYSIS OF RANKING RECEIVER AND A NON-LINEAR SQUARE LAW RECEIVER FOR FH-BFSK TRANSMISSION IN WORST-CASE PARTIAL BAND JAMMING

R. VISWANATHAN, K. TAGHIZADEH AND R. NERELLA
Department of Electrical Sciences and Systems Engineering
Southern Illinois University, Carbondale, IL 62901

ABSTRACT

The performance of linear and certain nonlinear combining square law receivers for FH-BFSK transmission under worst-case partial band jamming environment has been analyzed recently [1,2]. In this paper we investigate the performance of two alternate types of nonlinear combining methods. These methods of combining lead to a Minimum Rank Receiver (MRR) and a Product Combining (PCR) Receiver respectively. It is found that the PCR receiver, which does not require either the signal to thermal noise ratio or the signal to jamming ratio for its operation, performs nearly as good as the clipper receiver considered in [1].

INTRODUCTION

In this paper we consider an L hops per bit frequency hopped - binary frequency shift keying (FH-BFSK) system. Here, each bit is divided into L independent transmissions of τ seconds duration (1/L th of bit time T_b seconds) by means of a frequency hopping scheme. This suggests the hop rate R_H is L time the bit rate R_b. A PRBS generator selects one of N frequencies available in a frequency - synthesizer, every τ seconds. The selected tone is then modulated by the binary data using frequency shift keying. It is realized that the same data over T seconds will be modulating L different tones during that bit period. The diversity (when L>1) thus introduced may be advantageous to reduce the effect of intentional interference and thermal noise. At the receiver, after dehopping with an identical synchronized frequency-synthesizer, noncoherent energy detection is employed to detect the energy in either frequency f_1 (mark) or frequency f_2 (space) over each of τ seconds duration (Figure 1). The process is repeated L times to obtain 2L energy samples every T_b seconds. Depending on the type of combining scheme used to utilize the information in these 2L samples, we get different types of receivers.

When the samples are combined linearly, it is found that L>1 leads to poor performance [1,2]. In such a situation, noncoherent combining loss dominates and multi hops per bit is not useful. However, if the samples are passed through a soft limiter before the summing operation, we get a clipper receiver [1]. For clipper, it is found that moderate L values lead to less probability of error, especially for low signal to jamming ratios.

As in [1] we account for the presence of thermal noise fully. The noise is assumed flat with two sided power spectral density of $N_o/2$. It is assumed that the jammer has a total of J watts but chooses to jam a fraction of the transmission band for the purpose of effective jamming [1,3]. Under this condition, the jammer noise power in the jammed cell [see Figure 2] is

$$\sigma_J^2 = \frac{B}{\gamma W} J \qquad 0 < \gamma \leq 1 \qquad (1)$$

Each hopped tone is then jammed with probability γ or not jammed with probability $(1-\gamma)$.

In the following analysis, without loss of generality, let us assume that the space tone is transmitted over $0 < t < T_b$. The 2L samples can be arranged in a matrix form as shown below

space samples
(or)
correct row

$$\begin{pmatrix} y_{11} & y_{12} \cdots & y_{1L} \\ y_{21} & y_{22} \cdots & y_{2L} \end{pmatrix}$$

mark samples
(or)
incorrect row

The samples $y_{11}\ y_{12} \cdots y_{12}$ will then be due to the signal plus thermal noise and possible jamming noise whereas the samples $y_{21}\ y_{22} \cdots y_{2L}$ are due only to thermal noise and possible jammer power. Therefore, any receiver commits an error in the decision, if it decides the second row as the correct row. The above samples can be shown to have the following density functions [4]:

$$f(y_{1j}) = \frac{1}{2N_i} e^{-\frac{y_{1j}}{2N_i}} e^{-A^2/2N_i} I_o(\sqrt{y_{1j}}\, A/N_i) \qquad y_{1j}>0 \qquad (2a)$$

$$f(y_{2j}) = \frac{1}{2N_i} e^{-y_{2j}/2N_i} \qquad y_{2j}>0 \qquad (2b)$$

Here A denotes the amplitude of the received tone, i=1 denotes jamming and i=2 denotes no jamming. j takes values from 1, 2. .L. Within a normalizing constant, (2a) represents the non-central chi-squared distribution with two degrees of freedom.

It is assumed that when a tone slot is jammed, the adjacent FSK slot is also jammed. Therefore, this implies that for a particular j, both (2a) and (2b) have either N_1 (jammed) or N_2 (unjammed). Thus

$$\begin{aligned} N_1 &= B\,(N_0 + N_J/\gamma) \\ N_2 &= BN_0 \\ B &= 1/\tau \end{aligned} \qquad (3)$$

Let us also define the following:
signal bit energy to noise density ratio = (E_b/N_0)
signal bit energy to jamming density ratio = (E_b/N_J)

$$\frac{E_b}{N_0} = \frac{A^2 L\tau}{2N_0} = \frac{A^2 L/2}{N_2} \qquad (4)$$

$$\frac{E_b}{N_J} = \frac{A^2 L\tau}{2N_J} = \frac{A^2 L/2}{(N_1 - N_2)\gamma} \qquad (5)$$

$$R_1 = \frac{E_b/N_0}{E_b/N_J} \qquad (6)$$

$$\frac{N_1}{N_2} = 1 + R_1/\gamma \qquad (7)$$

$$\alpha_1 = A^2/N_1 \qquad (8)$$

$$\alpha_2 = A^2/N_2 \tag{9}$$

In the following sections, we investigate the performance of two types of nonlinear combining schemes. Because of the complexity of analysis, complete theoretical result has been possible only for L=2. Simulation has been carried out for L=4 case. First, we obtain the probability of error conditioned on the jamming state information. That is, ℓ=0 corresponds to none of the two slots being jammed, ℓ=1 to only one slot jammed and ℓ=2 to both slots being jammed. Then the average probability of bit error is given by

$$P(e \mid \gamma) = \sum_{\ell=0}^{L} P(e \mid \gamma, \ell)\, \gamma^{\ell}\, (1-\gamma)^{2-\ell} \quad \binom{L}{\ell} \tag{10}$$

Then by numerical computation, γ will be varied to locate the worst case P(e). That is,

$$P(e) = \max_{\gamma} P(e \mid \gamma) \tag{11}$$

RANK RECEIVER

Nonparametric type receivers like rank receiver has been found to be competitive to near optimum parametric receivers under multiple access interference conditions [5]. It can be easily shown that the samples in the correct row are stochastically larger than the samples in the incorrect row. This suggests the possibility of a rank receiver to combat worst-case partial band jamming.

When L equals 2, we are considering a total of four samples. In the ranking process, the largest sample among the 4 samples is assigned rank 4, the next largest the rank 3 and so on [6]. Therefore the ranks range from 1 to 4. All the possible rank assignments are shown in Figure 3. Each of the A_i's shown in the figure has 4 possible permutations. If we denote the ranks of the samples as R_{11} R_{12} R_{21} and R_{22}, we can formulate three possible rank receivers:

Maximum Rank Sum Receiver

Here the rank sums $S_i = R_{i1} + R_{i2}$, i = 1, 2 are computed and the row corresponding to the largest rank sum is chosen as the correct row.

Minimum Rank Receiver (MRR)

This receiver is based on the decision

$$\min(R_{11}, R_{12}) \underset{\text{row 2}}{\overset{\text{row 1}}{\gtrless}} \min(R_{21}, R_{22}) \tag{12}$$

Maximum Rank Receiver

This receiver is based on

$$\max(R_{11}, R_{12}) \underset{\text{row 2}}{\overset{\text{row 1}}{\gtrless}} \max(R_{21}, R_{22}) \tag{13}$$

Upon defining

$$P_2 = \text{Prob}\ (\max(y_{11}, y_{12}) < \max(y_{21}, y_{22})) \tag{14}$$

$$P_1 = \text{Prob}\ (\min(y_{11}, y_{12}) > \min(y_{21}, y_{22},)) \tag{15}$$

we have

$$P_2 = P(A_1) + P(A_2) + P(A_3) \tag{16}$$

$$P_1 = P(A_2) + P(A_5) + P(A_6) \tag{17}$$

The probability of error in the decision, conditioned on γ and the jamming state information ℓ, can be computed as follows.

In MRR, an error occurs in the decision corresponding to the occurrence of any of the events A_1 A_3 or A_4. Then

$$P(e|\gamma,\ell) = 1-P_1 \tag{18}$$

Similarly, for maximum rank sum receiver

$$P(e \mid \gamma,\ell) = \frac{P_2 + (1 - P_1)}{2} \tag{19}$$

When error rates are computed, based on the following discussion, it is found that the worst-case probability of error is always the least for a minimum rank receiver. Therefore, the other two rank receivers are not discussed any further. One possible explanation for the good performance of the MRR in relation to the other two is as follows. Especially when (E_b/N_J) is large, a smart jammer tries to make the fraction γ really small for effective jamming. Under this situation, a hit causes large sample values to appear in both the correct and incorrect rows. Therefore, a good receiver under this situation avoids using large samples in the decision process. This is exactly what happens in the MRR or the clipper receiver of [1].

Calculation of the Probability $(1-P_1)$

When only one slot is jammed, none of the two samples among the 4 samples would be identical. Without of loss of generality, let us assume that the first column of samples has been jammed. For the sake of convenience, let

$y_{11} \sim f_1$ where f_1 is given in (2a) with N_1
$y_{12} \sim f_2$ where f_2 is given in (2a) with N_2
$Y_{21} \sim f_3$ where f_3 is given in (2b) with N_1
$Y_{22} \sim f_4$ where f_4 is given in (2b) with N_2

Let the corresponding CDF's be denoted by F_1, F_2, F_3 and F_4. Define the new random variables

$$Z_1 = \text{Min } (f_1\ f_2) \tag{20}$$

$$S_1 = \text{Min } (f_3\ f_4) \tag{21}$$

$$P(e|\gamma,\ell=1) = P(Z_1<S_1)$$

$$= \int_0^\infty F_{z_1}(x)\, f_{s_1}(x)\, dx \tag{22}$$

It is straightforward to find Fz_1 and fs_1 in terms of $f_1 \ldots f_4$ [7]. Therefore,

$$P(e|\gamma,\ell=1) = \int_0^\infty [F_1(\chi) + F_2(\chi) - F_1(\chi)\, F_2(\chi)]\, \lambda_3 e^{-\lambda_3 \chi}\, d\chi \tag{23}$$

where

$$\lambda_3 = \frac{1}{2N_1} + \frac{1}{2N_2} \tag{24}$$

Evaluating each integrand separately we have

$$P(e|\gamma,\ell=1) = I_{\lambda_3} + II_{\lambda_3} - III_{\lambda_3} \tag{25}$$

In general, the three integrals can be defined as

$$I_\lambda = \int_o^\infty F_1(\chi)\, \lambda e^{-\lambda\chi}\, d\chi$$

$$II_\lambda = \int_o^\infty F_2(\chi)\, \lambda e^{-\lambda\chi}\, d\chi \tag{26}$$

$$III_\lambda = \int_o^\infty F_1(\chi)\, F_2(\chi)\, \lambda e^{-\lambda\chi}\, d\chi$$

Therefore in (25), the values of I, II and III evaluated at $\lambda=\lambda_3$ are used.

Expressing a non-central chi-squared CDF as a mixture of central chi-square CDF's [8], it is possible to evaluate I_λ and II_λ in closed form.

$$I_\lambda = \frac{e^{-\frac{\alpha_1}{2}\left[\frac{2N_1\lambda}{1+2N_1\lambda}\right]}}{1+2N_1\lambda} \tag{27}$$

$$II_\lambda = \frac{e^{-\frac{\alpha_2}{2}\left[\frac{2N_2\lambda}{1+2N_2\lambda}\right]}}{1+2N_2\lambda} \tag{28}$$

In [9], an expression for III_λ has been found in terms of Marcum-Q functions. We have

$$III_\lambda = J_1 + J_2 \tag{29}$$

with

$$J_1 = K_1\left[\frac{C^2}{1+C^2}\left(1 - Q(\sqrt{b}, \sqrt{a})\right) + \frac{1}{1+C^2}\, Q(\sqrt{a}, \sqrt{b})\right] \tag{30}$$

$$K_1 = \frac{e^{-\frac{A^2}{2N_2}\,(1 - 1/(2\lambda N_2 + 1))}}{(2\lambda N_2 + 1)}$$

$$a = \frac{a_2^2}{\sigma_1^2 + \sigma_2^2}, \qquad b = \frac{a_1^2}{\sigma_1^2 + \sigma_2^2}, \qquad c = \frac{\sigma_1}{\sigma_2}$$

$$\sigma_1^2 = 2N_1\lambda + \frac{N_1}{N_2}, \qquad \sigma_2^2 = 1$$

$$a_1^2 = A^2\,(2\lambda + \frac{1}{N_2}), \qquad a_2^2 = \frac{A^2}{N_2\,(2\lambda N_2 + 1)}$$

J_2 is obtained from J_1 by interchanging N_1 and N_2.

When both slots are jammed, the samples y_{11} and y_{12} would be identical with density f_1 and the samples y_{21} would be identical with density f_3. Proceeding along similar lines, as done in case (i) above, we obtain

$$P(e \mid \gamma, \ell=2)$$

$$= \frac{2}{3}\, e^{-\alpha_1/3} - \frac{2}{3}\, e^{-\frac{\alpha_1}{3}} \left[\frac{3}{4} - \frac{3}{4}\, Q\left(\sqrt{\frac{3\alpha_1}{4}}, \sqrt{\frac{\alpha_1}{12}} \right) + \frac{1}{4}\, Q\left(\sqrt{\frac{\alpha_1}{12}}, \sqrt{\frac{3\alpha_1}{4}},\right) \right] \qquad (31)$$

An expression for $P(e|\gamma,\ell=0)$ is obtained from (31) by repacing α_1 with α_2.

Therefore, by using the expression for conditional error probabilities, the average probability of error given in (10) is computed. Different average error rates are obtained as a function of γ for each (E_b/N_0) and (E_b/N_J) in order to locate the worst-case error rate. The worst case probability of error for MRR is shown in Figure 4. For comparison purposes we also plot the error rate for a clipper receiver from [1]. In both the plots, E_b/N_0, is fixed at 13.35dB and the error rate is shown a function of (E_b/N_J). It is seen that the clipper receiver performs better than MRR, when L=2. However, the clipper receiver needs threshold adjustment as a function of (E_b/N_0) whereas MRR needs no adjustments. However, comparing with figures in [1], it is seen that MRR is better than a linear square law combiner. General analysis of MRR for any L seems difficult.

PRODUCT COMBINING RECEIVER (PCR)

The product combiner is merely the result of a trial procedure. Any statistical theory which might suggest its use under worst-case partial band jamming seems unavailable. It is known that the linear combiner is nearly the minimum probability of error receiver, when there is only wide-band jamming and low signal to noise ratio [4]. However, the product combiner is found to perform favorably as the theoretical results derived below shows. The product receiver chooses row 1 as the correct row when the product $y_{11}\, y_{12} \,..\, y_{12}$ is greater than the product $y_{21}\, y_{22} \,..\, y_{22}$ and chooses row 2 when the converse is true.

Error rate analysis when L=2

(i) One slot jammed (ℓ=0). Let the densities f_1, f_2, f_3 and f_4 be defined as in Section II.

Therefore,

$$P(e \mid \gamma, \ell=1) = 1 - \tilde{P} \qquad (32)$$

where

$$\tilde{P} = \text{Prob}\,(\, f_1\, f_2 > f_3\, f_4) = P(x > \frac{1}{y}) \qquad (33)$$

$$x = f_1/f_3 \qquad (34)$$

$$y = f_2/f_4$$

In statistical literature x and y are said to be distributed as non-central F [8].

Hence,

$$P(e|\gamma,\ell=1) = \int_0^\infty F_y\,(1/\chi)\; f_\chi(\chi)\; d\chi \tag{35}$$

We can express $F_y\,(y)$ as [8]

$$F_y\,(y) = \frac{y}{1+y}\; e^{-\frac{\alpha_2}{2}\left(\frac{1}{1+y}\right)} \qquad y > 0 \tag{36}$$

Then by using a series expansion for $f_x\,(\;)$ we can evaluate (35) as

$$P(e \mid \gamma,\ \ell=1) = \sum_{j=0}^{\infty} \frac{e^{-\frac{\alpha_1}{2}}\left(\frac{\alpha_1}{2}\right)^j}{j!\; B(1+j,\ 1)} \left[\int_0^\infty \frac{e^{-\alpha_2\frac{x}{2(1+x)}}}{(x+1)^{3+j}}\; x^j dx\right] \tag{37}$$

upon evaluating the integral,

$$P\,(e \mid \gamma,\ \ell=1) = (1 + \delta_1)\,\frac{e^{-\delta_2}}{\delta_2} + (FA1 - FA2) \tag{38}$$

where $\delta_1 = \frac{\alpha_1}{2}$ and $\delta_2 = \frac{\alpha_2}{2}$.

$$FA_1 = \frac{e^{-(\delta_1+\delta_2)}}{\delta_2}\; G \tag{39}$$

$$G = \frac{\delta_2^2}{(\delta_2 - \delta_1)^2}\left(e^{\delta_2} + e^{\delta_1}\left(\frac{\delta_1^2}{\delta_2} - \delta_1 - 1\right)\right) \tag{40}$$

$$FA_2 = \frac{e^{-(\delta_1+\delta_2)}}{\delta_2^2}\left(G + \delta_1\,\frac{dG}{d\delta_1}\right) \tag{41}$$

Proceeding along similar lines, we have

$$P\,(e \mid \gamma,\ \ell=2) = \frac{3 + \delta_1}{6}\; e^{-\delta_1} \tag{42}$$

$$P\,(e \mid \gamma,\ \ell=0) = \frac{3 + \delta_2}{6}\; e^{-\delta_2} \tag{43}$$

Using the above conditional probabilities and (10) the average error rate P(e) is evaluated. The results are shown in Figure 4. It is seen that for L=2, the Product Combining Receiver is nearly as good as the clipper receiver. Whereas the clipper receiver is of adaptive type, the PCR needs no adaptation. Using IMSL routines GGExP and GGNML to generate the exponential and Gaussian samples, we simulated the performance of PCR for L=4. Worst-case P(e) was estimated for three (E_b/N_J) values of 15, 20 and 25dB and for fixed (E_b/N_0) value of 13.35dB. The results are shown in Table 1. As observed from simulation, the PCR is only slightly inferior to a clipper receiver for this case.

CONCLUSION

In this paper we analyzed two types of nonlinear combining receivers for L hops per bit FH-BFSK transmission under worst-case partial band interference. The minimum rank receiver was found to perform poorly compared to the clipper receiver suggested in [1]. However, the other combining receiver, termed as PCR, is nearly as good as the clipper receiver. Both the receivers considered in the paper are simple to implement and they do not need any adaptation. This is different from the clipper, which needs threshold adaption.

Theoretical study is limited to L=2 situation and limited simulation study is done for L=4. For general value of L, Chernoff bounding technique may be used, but the tightness of the bound seems questionable.

REFERENCES

1. Johng S. Lee, Leonard E. Miller and Young K. Kim. Error performance analysis of linear and non-linear combining square law receivers for L hops per bit FH-BFSK waveforms in worst-case partial band jamming. 1983 IEEE Military Communication Conference, October 31 - November 2, 1983, Washington D.C., Conference Record, pp. 22-28.
2. Y. K. Kim, L. E. Miller, J. S. Lee. (1983) The exact performance analysis of two types of adaptive receivers for multihops per symbol FH-MFSK systems in partial band noise jamming and system thermal noise. IEEE Global Communication Conference, pp. 38.2.1-38.2.6.
3. A. J. Viterbi and I. M. Jacobs. (1975) Advances in coding and modulation for non-coherent channels affected by fading, partial band and multiple access interference. Advance in Communication Systems, Vol 4, New York: Academic Press, Inc., A. J. Viterbi (ed.), pp. 279-308.
4. A.D. Whalen. (1971) Detection of signals in noise. Academic Press.
5. R. Viswanathan and S. C. Gupta. Nonparametric receivers for FH-MFSK mobile radio. IEEE Transaction on Communication, to be published.
6. J. D. Gibbons. (1971) Nonparametric statistical interference. McGraw-Hill Book Co., New York.
7. A. M. Mood, F. A. Graybill, D. C. Boes. (1974) Introduction to the theory of statistics. McGraw-Hill Book Co., New York.
8. N. L. Johnson and S. Kotz. (1970) Distribution in statistics - continuous univariate distribution-2. Houghton Mifflin Company, Boston.
9. R. Viswanathan and K. Taghizadeh. Probability of maximum of two noncentral chi-square variate exceeding an exponential variate. Unpublished report.
10. M. Abramowitz and I. Stegan. (1965) Handbook of mathematical functions. New York, Dover.

Table 1. FH-BFSK PCR Performance

$\left(\frac{E_b}{N_0}\right)$ dB	P_b, PCR (Simulation)	P_b, Clipper from Ref. [1]
15	8.5×10^{-3}	6.5×10^{-3}
20	1.6×10^{-3}	1.15×10^{-3}
25	7.8×10^{-4}	4.2×10^{-4}

L = 4 hops/bit, $\left(\frac{E_b}{N_0}\right)$ = 13.35 dB

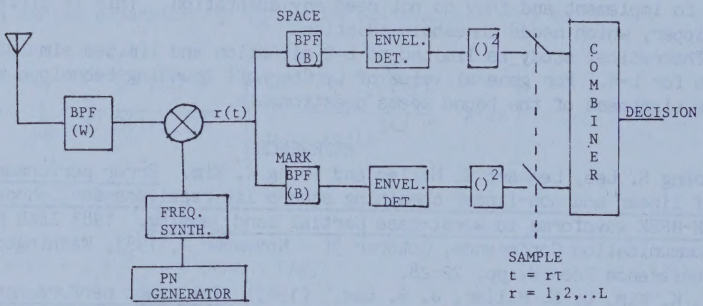


FIG. 1 RECEIVER FOR FH/BFSK SYSTEM

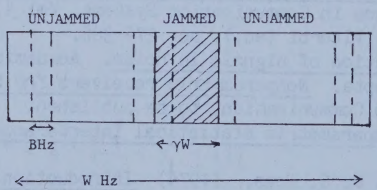


FIG 2.
PARTIAL BAND JAMMING MODEL

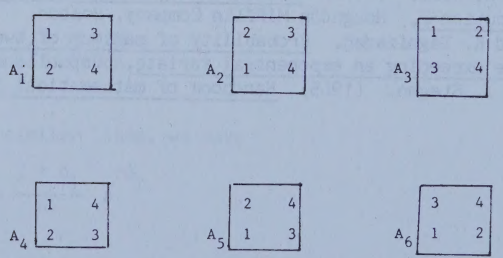


FIG. 3 POSSIBLE RANK ASSIGNMENTS

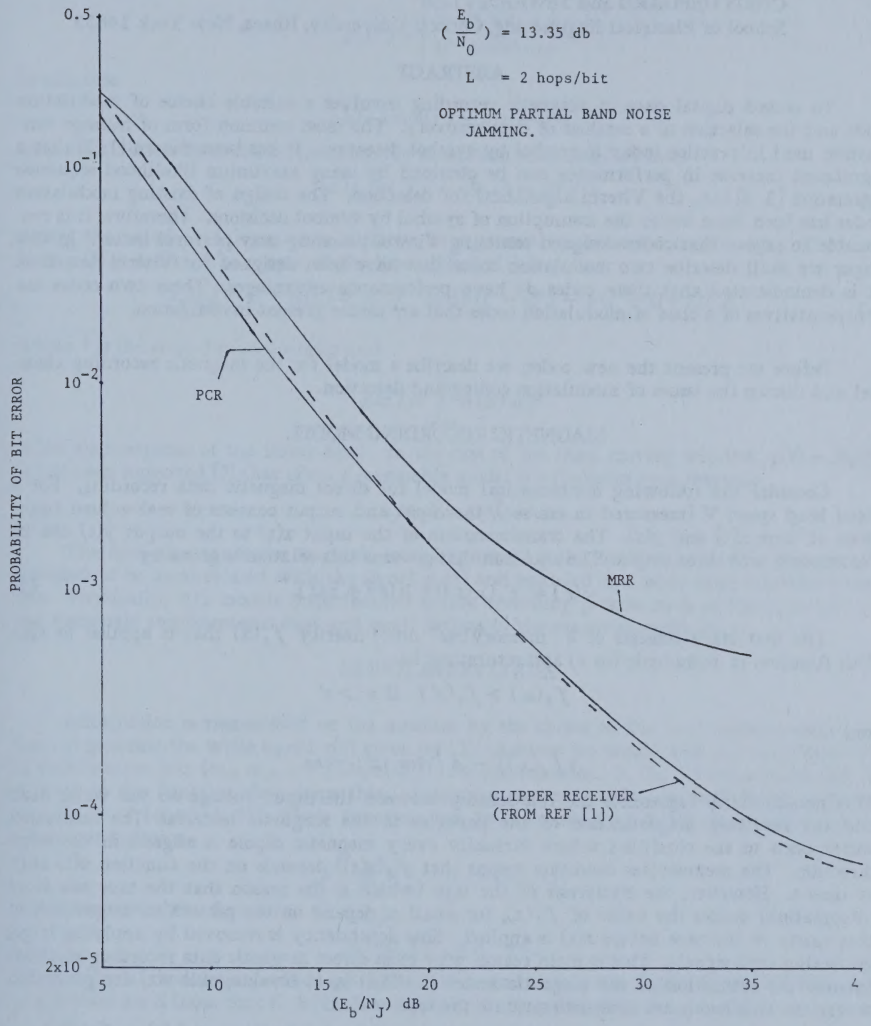


FIG 4 FH-BFSK RECEIVER PERFORMANCE COMPARISON

NEW MODULATION CODES FOR MAGNETIC RECORDING

CHRIS HEEGARD and TING-ANN LEE
School of Electrical Engineering, Cornell University, Ithaca, New York 14853

ABSTRACT

To record digital data in magnetic recording involves a suitable choice of modulation code and the selection of a method of data recovery. The most common form of message estimation used in practice today is symbol by symbol detection. It has been shown [1, 2] that a significant increase in performance can be obtained by using maximum likelihood sequence estimation [3, 4] (i.e., the Viterbi algorithm) for detection. The design of existing modulation codes has been done under the assumption of symbol by symbol decisions. Therefore, it is reasonable to expect that codes designed assuming Viterbi decoding may perform better. In this paper we shall describe two modulation codes that have been designed for Viterbi detection. It is demonstrated that these codes do have performance advantages. These two codes are representatives of a class of modulation codes that are under present investigation.

Before we present the new codes, we describe a model for the magnetic recording channel and discuss the issues of modulation coding and detection.

MAGNETIC RECORDING MODEL

Consider the following mathematical model for direct magnetic data recording. For a fixed head speed V (measured in cm/sec), the input and output consists of real-valued functions of time x(t) and y(t). The transformation of the input x(t) to the output y(t) can be decomposed into three stages. The equation that governs this relation is given by

$$y(t) = f_t(x(t)) * h(t) + z(t). \tag{1}$$

The first stage consists of a "memoryless" non-linearity $f_t(x)$ that is applied to x(t). This function is monotonic (in x) and saturating, i.e.

$$f_t(x) > f_t(x') \quad \text{if } x > x'$$

and

$$|f_t(x)| \to A \quad \text{for } |x| \to \infty$$

This non-linearity represents the relationship between the input voltage on the write head and the resulting magnetization of the particles in the magnetic material. The saturation corresponds to the condition where virtually every magnetic dipole is aligned in the same direction. The memoryless condition means that $f_t(x(t))$ depends on the function x(t) only at time t. However, the hysteresis of the tape (which is the reason that the tape can store information) makes the value of $f_t(x)$, for small x, depend on the pattern of magnetization that exists on the tape before x(t) is applied. This dependency is removed by applying large, saturating read signals. This is main reason why most direct magnetic data recording methods involve the saturation of the magnetic material. That is, bi-level signals x(t) are generated where the two levels are chosen to saturate the tape or disk.

The second stage of the recording process consists of a filtering of $\tilde{x}(t) = f(x(t))$ by a linear time-invariant filter with impulse response h(t). This filter corresponds to a differentiation and low-pass filtering of $\tilde{x}(t)$. The differentiation arises from the fact that the output voltage of the read head is induced by a changing magnetic field (Faraday's law). This

This research was supported by NSF Grants ECS-8204886 and ECS-835220 and the Eastman Kodak Co.

induction is a result of the motion of the magnetic material under the read head. The low-pass characteristic is a result of the fact that the read head has a small positive gap size. This gap results in a moving average of the signal. For example, an ideal moving window would be a box function

$$B_\tau(t) = \begin{cases} 1 & 0 \leqslant t < \tau \\ 0 & otherwise \end{cases} \tag{2}$$

In this case

$$h(t) = \delta(t) - \delta(t - \tau)$$

where $\delta(t)$ is the dirac delta function. For saturation recording, we take

$$x(t) = \sum_i \alpha_i B_T(t - iT) \tag{3}$$

where the $\{\alpha_i \in \{-1, 1\}\}$ sequence is chosen to represent the recorded information. Then we have $\tilde{x}(t) = A\, x(t)$. The output of the linear filter h(t) is

$$x_o(t) = \tilde{x} * h(t) = A \sum_i \alpha_i [p(t - iT) - p(t - (i+1)T)] \tag{4}$$

where * is the convolution operator and

$$p(t) = \int_{-\infty}^{t} h(\sigma)\, d\sigma$$

is the step response of the linear filter. In the case of the ideal moving window, $p(t) = B_\tau(t)$. It has been suggested [5] that often a reasonable model is a Gaussian type response

$$p(t) = e^{-t^2}.$$

The final element to the magnetic recording model is the additive noise z(t). The noise is assumed to be uncorrelated with the signal $x_o(t)$ and modeled as a wide sense stationary process. Physically, z(t) models imperfections in the recording process such as thermal noise in the electronic amplifiers and dust and small defects in the magnetic medium.

MODULATION CODES

Information is represented on the medium by the choice of the $\{\alpha_i\}$ sequences that are used to generate the write signal x(t) given by (2). Assume we would like to store a sequence of information bits $\{m_1, m_2, \cdots\}$ $(m_i \in \{0, 1\})$ on the recorder. A (K, N) modulation code is defined as the function that maps this information sequence onto the modulation sequence $\{\alpha_1, \alpha_2, \cdots\}$ $(\alpha_i \in \{-1, 1\})$. These functions can be described by a finite state machine with an output function

$$f: \{0, 1\}^K \times S \rightarrow \{-1, 1\}^N$$

and a state transition function

$$g: \{0, 1\}^K \times S \rightarrow S.$$

Thus, a (K, N) modulation code maps K bits of data and a state $s \in S$ onto N code symbols and a new state $s \in S$ (note that $K \leqslant N$). For example, if the initial state is $s_0 \in S$, then

$$\alpha_1, \alpha_2, \cdots, \alpha_N = f(m_1, m_2, \cdots, m_K, s_0)$$

and

$$s_1 = g(m_1, m_2, \cdots, m_K, s_0).$$

The rate of such a modulation code is measured by the ratio R = K/N (measured in bits per

symbol). For a fixed head speed V and symbol period T, this results in the recording of R/T bits per second or R/(VT) bits per centimeter.

There are many examples of modulation codes that are used in practice [6-11]. The most common of these are NRZ (non-return to zero) and NRZI (non-return to zero, inverse). These are examples of codes with K = N = 1. For NRZ, the sign of α_i is directly related to the value of m_i (positive for 1 and negative for 0). In this case, the state space S is trivial and

$$\alpha_i = f(m_i) = 2m_i - 1.$$

In NRZI, the value of the product $\alpha_i \alpha_{i-1}$ tells us the value of m_i (positive for 0 and negative for 1). For NRZI, the state space S = {0, 1},

$$\alpha_i = f(m_i, w_{i-1}) = 2(m_i + w_{i-1})_2 - 1.$$

and

$$w_i = g(m_i, w_{i-1}) = (m_i + w_{i-1})_2$$

where $(m_i + w_{i-1})_2$ represents the modulo 2 sum. Thus, for NRZ a transition in x(t) occurs whenever a transition occurs in the data, while in NRZI, a transition in x(t) corresponds to a 1 in the message. These codes are examples of finite state encoders which are represented by linear filtering (or precoding) of the data over the finite field F_2 [5]. Note that if we define the D-transform of a sequence, e.g.,

$$m(D) = \sum_{i=0}^{\infty} m_i D^i$$

we can describe the filtering by the equation

$$w(D) = (m(D) H(D))_2$$

where H(D) is the D-transform of the unit sample response of the filter. Then the modulation sequence is given by

$$\alpha(D) = 2w(D) - u(D)$$

where

$$u(D) \equiv \sum_{i=0}^{\infty} D^i.$$

For NRZ,

$$H(D) = 1$$

and for NRZI

$$H(D) = \frac{1}{1-D} = u(D).$$

It has been suggested [5] that a possible precoding for high density recording might be

$$H(D) = \frac{1}{1-D^2} = u(D^2) = \sum_{i=0}^{\infty} D^{2i}.$$

This has been termed "Interleaved NRZI" (INRZI). Note that this encoder has 4 states since

$$w_i = g(m_i, w_{i-1}, w_{i-2}) = (m_i + w_{i-2})_2$$

Modified Frequency Modulation (MFM, also known as Miller code or delay modulation [7, 8]) is an example of a (K = 1, N = 2) code. In terms of a finite state machine description, S = {0, 1}x{−1, 1}, the state is the pair $s_i = (m_{i-1}, \alpha_{2i-1})$,

$$\alpha_{2i} = \begin{cases} \alpha_{2i-1} & \text{if } m_i = 1 \text{ or } m_{i-1} = 1 \\ -\alpha_{2i-1} & \text{if } m_i = m_{i-1} = 0 \end{cases}$$

and

$$\alpha_{2i+1} = \alpha_{2i}(1 - 2m_i).$$

We can see that transitions only occur between α_{2i} and α_{2i+1} when $m_i = 1$ and between α_{2i-1} and α_{2i} when $m_{i-1} = m_i = 0$. This is one example of several (K = 1, N = 2) codes that have been developed to introduce transitions and to minimize the power spectrum at dc.

Other classes of modulation codes that have been developed are called run-length codes [9-11]. These codes are parameterized by two nonnegative integers [d, k] ($0 \leq d < k \leq \infty$) that represent run-length constraints on the modulation codewords. All the codewords satisfy the condition that after a transition (e.g., $\alpha_1 = +1$, $\alpha_2 = -1$) there will not be a transition for at least d clocks (e.g., $\alpha_i = -1$, i = 3, 4, ... , 2+d) and there will be a transition within k clocks (e.g., $\alpha_i = +1$, for some i = 3+d, ..., 2+k). One such run-length code that has been used in actual systems is a [d = 2, k = 7] code [9-11]. This code is generated by a [K=1, N=2] encoder. Another popular run-length code is called 3PM [11], it has run length constraints [d = 2, k = 11] and is also described by an [K = 1, N = 2] modulation encoder.

DETECTION

The recovery of information involves the detection of the message sequence $\{m_i\}$ from the read signal y(t) given by (1), (3) and (4). For saturation recording,

$$y(t) = x_o(t) + z(t)$$

where

$$x_o(t) = A \sum_{i=0}^{\infty} \alpha_i \, [p(t - iT) - p(t - (i+1)T)].$$

To detect $\{m_i\}$, we filter y(t) with a linear time invariant filter with impulse response g(t) (called a linear equalizer) and sample every T seconds

$$y_i = g^* y(iT) = x_i + z_i$$

where

$$x_i = g^* x_o(iT) = A \sum_{k=0}^{\infty} \alpha_{i-k} \, a_k .$$

The coefficients of the linear filter

$$a_k = g^* p(kT) - g^* p((k-1)T)$$

are samples of the overall response to the input B_T(t) and

$$z_i = g^* z(iT)$$

are samples of the filtered noise process. Thus we see that the $\{y_i\}$ sequence consists of a linearly filtered version of the $\{\alpha_i\}$ sequence corrupted by the (possibly) colored noise sequence $\{z_i\}$. In terms of the D-transform,

$$y(D) = x(D) + z(D), \quad x(D) = A \, a(D)\alpha(D). \tag{5}$$

The most common form of message detection used in practice today is symbol by symbol detection. In this case, the equalizer is chosen so that a reliable estimate $\hat{x}_i$ of x_i is made based solely on the value of the sample y_i. Then a mapping onto the message sequence is performed. Note that this mapping may not correspond to a simple inversion since under symbol by symbol detection, the estimated sequence $\{\hat{x}_i\}$ may not correspond with a valid sequence $\{x_i\}$ (e.g.,

for a run-length limit code, the estimated sequence A $\hat{\alpha}(D) = \hat{x}(D)/\ a(D)$ may not satisfy the [d, k] constraint). This can lead to a propagation of errors (i.e. a single error in detecting x_i leads to several message errors). For symbol by symbol detection, the choice of the equalizer filter g(t) represents a tradeoff between the noise severity (i.e. signal to noise ratio) and intersymbol interference (i.e., the signal to distortion ratio).

The second type of detector tries to estimate the sequence $\{m_i\}$ directly from the sequence $\{y_i\}$. With a proper choice of g(t), this problem can be formulated as trying to estimate the outcome of a finite state Markov process (e.g., a finite state machine driven by random inputs) observed through noise. If this is the case, then we can apply a maximum likelihood procedure, known as the Viterbi algorithm, to this problem [3, 4]. We note from (5), that if a(D) is a polynomial of degree r, then the x_i sequence can be considered as the output of a finite state machine with input α_i and state $(\alpha_{i-1}, \alpha_{i-2}, \ldots, \alpha_{i-r})$. For saturation recording, $\alpha_i \in \{-1, 1\}$, implies that the number of states is 2^r (assuming $a_0 \neq 0$). Since the channel inputs α_i are themselves the product of a finite state machine with input sequence m_i, the encoder/channel cascade is finite state machine with input m_i and output x_i.

THE NEW CODES

For purposes of comparison, we shall use the notion of "effective" signal to noise ratio. As we shall briefly demonstrate, we can often approximate the bit error probability of a modulation code and detector by

$$P_e \approx constant\ Q(\sqrt{SNR_e})$$

where SNR_e is the effective signal to noise ratio and

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-z^2/2}\, dz.$$

This SNR_e is of the form of a product

$$SNR_e = SNR_s\, d^2_{\min}$$

where SNR_s is the "symbol" signal to noise ratio and $d^2_{\min}$ is the minimum number of symbols that distinguish messages at the detector ($d^2_{\min}$ is a normalized minimum distance).

Assume for simplicity that the step response of the magnetic channel is the ideal moving average given by (2) and that the noise process z(t) is white, zero-mean Gaussian with (two-sided) spectrum σ_z^2. We shall take the equalizer response to be

$$g(t) = \begin{cases} \frac{1}{2AT} B_T(-t) & \text{if } T \leqslant \tau \\ \frac{1}{2A\tau} B_\tau(-t) & \text{if } T \geqslant \tau \end{cases}.$$

(Note that causal filters can be obtained by the introduction of finite delay.) This choice of response guarantees that the sampled process z_i is independent, identically distributed Gaussian noise with variance $\sigma_z^2/\min\{T, \tau\}4A$.

We shall consider two cases. First, if $T = \tau$ (or $T \geqslant \tau$) then $x_i = (\alpha_i - \alpha_{i-1})/2$ or $a(D) = (1 - D)/2A$. In the case of NRZI precoding, $H(D) = 1/1\text{-}D$, the usual symbol by symbol rule is given by

$$\hat{m}_i = \begin{cases} 0 & \text{if } |y_i| < 1/2 \\ 1 & \text{if } |y_i| \geqslant 1/2 \end{cases}. \tag{6}$$

This rule has a bit error probability (for equally likely data)

$$P_e = \frac{3}{2}Q(\sqrt{\rho}) - \frac{1}{2}Q(3\sqrt{\rho}) \approx \frac{3}{2}Q(\sqrt{\rho}),$$

where

$$\rho = \frac{4A^2\tau}{\sigma_z^2} \tag{7}$$

is the "symbol energy to noise ratio". Thus the effective signal to noise ratio $SNR_e = \rho$ in this case; the minimum energy used to distinguish signals corresponds to one transition or $d^2_{\min} = 1$. It has been shown [3, 4] that, for Viterbi detection, the bit error probability

$$P_e \approx 4\,Q(\sqrt{2\rho}).$$

In terms of effective signal to noise ratio $SNR_e = 2\rho$, an improvement of 3 dB. This improvement is related to that fact that the minimum energy (finite duration) error event in the Viterbi decoder corresponds to two transitions or $d^2_{\min} = 2$.

The other case of interest is when $T = \tau/2$. Then $x_i = (\alpha_i - \alpha_{i-2})/2$ or $a(D) = (1 - D^2)/2A$. Under the usual symbol by symbol rule (6) with INRZI precoding, $H(D) = 1/(1-D^2)$, the error probability

$$P_e \approx \frac{3}{2}Q(\sqrt{\rho/2}),$$

where ρ is given by (7). This expression reflects the fact that the symbol signal to noise ratio, SNR_s, is 3dB less than the previous case ($SNR_e = SNR_s = \rho/2$, $d^2_{\min} = 1$). This 3dB loss can of "recovered" by Viterbi decoding since as previously discussed this method has $d^2_{\min} = 2$ (i.e., INZRI with Viterbi detection has $SNR_e = \rho = 2SNR_s$).

The codes that we present involve (K, N) modulation encoders in which the state of the encoder consists of L past data inputs and M past channel inputs (i.e., $S = \{0, 1\}^L \times \{-1, 1\}^M$). The state transitions are described by

$$s_i = (m_{i-1}, \cdots, m_{i-L}, \alpha_{Ni-1}, \cdots, \alpha_{Ni-M})$$

and

$$s_{i+1} = (m_i, \cdots, m_{i-L+1}, \alpha_{(N+1)i-1}, \cdots, \alpha_{(N+1)i-M}).$$

The encoder output for k = 0, 1, ..., N-1 are described by the N functions

$$\alpha_{Ni+k} = f_k(m_i, s_i).$$

We present here two examples of K = 1 and N = 2 (rate 1/2) codes. For the NRZI channel, $a(D) = (1 - D)/2A$, consider the code with state (m_{i-1}, α_{2i-1}) and

$$\alpha_{2i} = (1 - 2m_i)(1 - 2m_{i-1})\alpha_{2i-1}$$

$$\alpha_{2i+1} = 1 - 2m_{i-1}.$$

This code has 4 encoder states and 4 decoder states (the number of decoder states is determined by the number of states of the encoder/channel cascade). This code has $d^2_{\min} = 4$ and thus an effective signal to noise ratio $SNR_e = 4\rho$. This code is described by the following table:

m_i	m_{i-1}	α_{2i-1}	α_{2i}	α_{2i+1}	x_{2i}	x_{2i+1}
0	0	+1	+1	+1	0	0
1	0	+1	-1	+1	-1	+1
0	1	+1	-1	-1	-1	0
1	1	+1	+1	-1	0	-1
0	0	-1	-1	+1	0	+1
1	0	-1	+1	+1	+1	0
0	1	-1	+1	-1	+1	-1
1	1	-1	-1	-1	0	0

This code improves the probability of error by increasing SNR_e by 3 db over uncoded NRZI with Viterbi decoding (6 db over symbol by symbol detection). The cost is a 1/2 reduction in the density of the data. This importance of this code is that it demonstrates how to make this tradeoff. This next code is more interesting. It maintains the same data density as uncoded NRZI yet improves the error probability.

For the INRZI channel, a(D) = (1 - D^2)/2A, consider the code with state (m_{i-1}, α_{2i-1}, α_{2i-2}) and

$$\alpha_{2i} = (1 - 2m_i)[(1 - 2m_{i-1})(\alpha_{2i-1}+\alpha_{2i-2}) - (\alpha_{2i-1}-\alpha_{2i-2})]/\ 2$$

$$\alpha_{2i+1} = (1 - 2m_i)[(1 - 2m_{i-1})(\alpha_{2i-1}-\alpha_{2i-2}) + (\alpha_{2i-1}+\alpha_{2i-2})]/\ 2$$

This code has 8 encoder and 8 decoder states and an $SNR_e = 3\rho$ ($d^2_{\min} = 6$). It is described by the following table:

m_{i-1}	m_i	α_{2i-2}	α_{2i-1}	α_{2i}	α_{2i+1}	x_{2i}	x_{2i+1}
0	0	+1	+1	+1	+1	0	0
0	1	+1	+1	-1	-1	-1	-1
1	0	+1	+1	-1	+1	-1	0
1	1	+1	+1	+1	-1	0	-1
0	0	+1	-1	+1	-1	0	0
0	1	+1	-1	-1	+1	-1	+1
1	0	+1	-1	+1	+1	0	+1
1	1	+1	-1	-1	-1	-1	0
0	0	-1	+1	-1	+1	0	0
0	1	-1	+1	+1	-1	+1	-1
1	0	-1	+1	-1	-1	0	-1
1	1	-1	+1	+1	+1	+1	0
0	0	-1	-1	-1	-1	0	0
0	1	-1	-1	+1	+1	+1	+1
1	0	-1	-1	+1	-1	+1	0
1	1	-1	-1	-1	+1	0	+1

Note that this code has the same data density as uncoded NRZI yet has a better SNR_e. The "coding gain" in this case (i.e. the ratio of the SNR_e of the coded INRZI channel to the SNR_e of the uncoded NRZI channel) is 3/2 or 1.8 db. The graph (figure 1) shows the result of a computer simulation demonstrating the improvement in bit error rate for this code.

These two codes are simple examples of modulation codes that are designed to conserve signal to noise ratio by maximizing $d^2_{\min}$. Higher values of $d^2_{\min}$ are possible by increasing the

number of states of the encoder and decoder. The following issues are under present investigation: (1) finding systematic ways of finding good codes, (2) obtaining bounds on possible performance of such codes, (3) determining the minimum number of decoder states for a given encoder/channel cascade, (4) determining the role of precoding on coded systems.

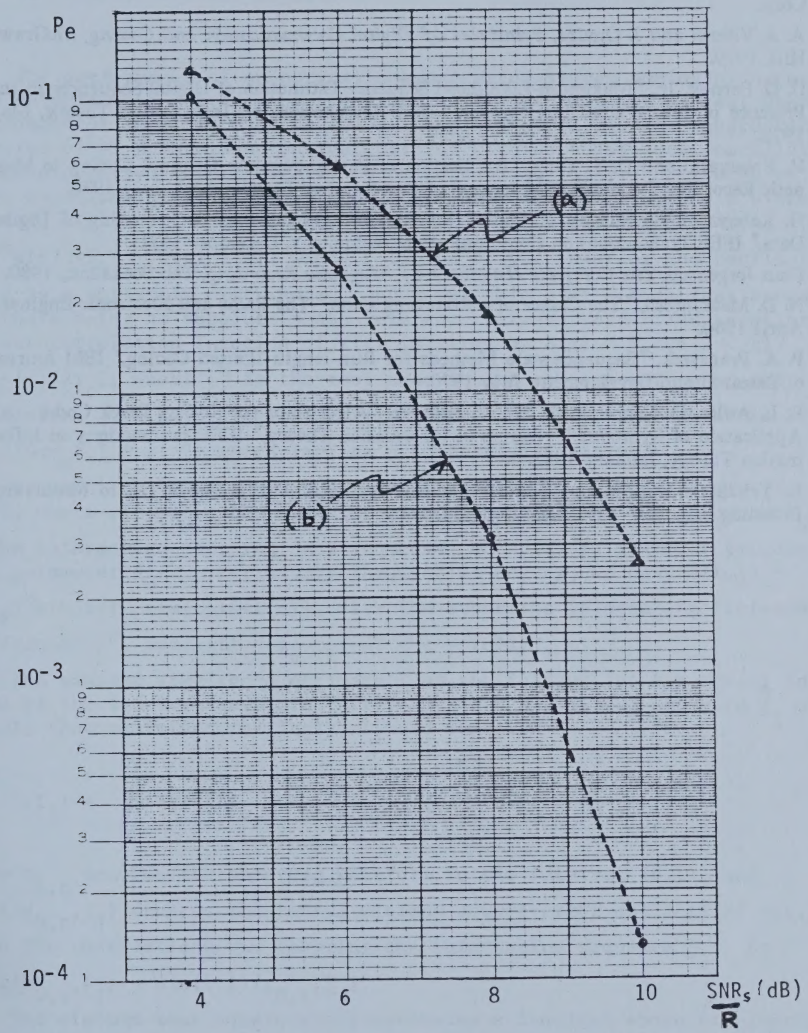


Figure 1: (a) Uncoded NRZI, (b) Rate 1/2, Coded INRZI.

REFERENCES

[1] H. Kobayashi, "Application of Probabilistic Decoding to Digital Magnetic Recording Systems," IBM Journal of Research and Development, Volume 15, pp. 64-75, January 1971.

[2] Roger Wood, "Viterbi Reception of Miller Squared Code on a Tape Channel," Ampex Corp.

[3] A. J. Viterbi and J. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979.

[4] G. D. Forney, Jr., "Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference," IEEE Transactions on Information Theory, May 1972.

[5] H. Kobayashi and D. T. Tang, "Application of Partial-response Channel Coding to Magnetic Recording Systems," IBM Journal of Research and Development, July 1970.

[6] H. Kobayashi, "A Survey of Coding Schemes for Transmission or Recording of Digital Data," IEEE Transactions on Communication Technology, December 1971.

[7] Finn Jorgenson, *The Complete Handbook of Magnetic Recording*, Tab Books Inc., 1980.

[8] N. D. Mackintosh, "The Choice of a Recording Code," The Radio and Electronic Engineer, April 1980.

[9] P. A. Franaszek, "Sequence-state Methods for Run-length-limited Coding," IBM Journal of Research and Development, July 1970.

[10] R. L. Adler, D. Coppersmith and M. Hassner, "Algorithms for Sliding Block Codes - An Application of Symbolic Dynamics to Information Theory," IEEE Transactions on Information Theory, January 1983.

[11] L. Yencharis, applications editor, "Small Winchester Drives Move Up to Mainframe Encoding Schemes," Electronic Design, October 15, 1981.

A HETERODYNE RECEIVER FOR AN OPTICAL COMMUNICATION SYSTEM WITH AN UNSTABLE LASER LOCAL OSCILLATOR*

COSTAS N. GEORGHIADES and DONALD L. SNYDER
Washington University
St. Louis, Missouri 63130

The performance of an optical communication system employing heterodyne detection is degraded by phase instabilities that are present when a semiconductor laser is used as a local oscillator [1,2]. The postdetection spectrum of an unmodulated optical carrier with unstable local oscillator has been observed experimentally to be Lorentzian [3,4]. Motivated by these experimental results, we model the phase instabilities by a Wiener process, $w(t)$, so the oscillator output signal $z(t)$ is given by $z(t) = (2P)^{\frac{1}{2}}\sin[2\pi f_0 t + w(t) + \theta]$, where P is the output power of the oscillator, and θ is a uniformly distributed random variable over $(-\pi,\pi)$ that accounts for there being no absolute phase reference between sender and receiver. The autocorrelation function of $z(t)$ is given by $R(\tau) = P\cos(2\pi f_0\tau)\times \exp[-|\tau|/2\tau_c]$, where $\tau_c^{\frac{1}{2}}$ is the root mean-square coherence-time of the oscillator. The corresponding power spectrum of $z(t)$ at baseband is $S(f) = 4P\tau_c[(4\pi\tau_c f)^2 + 1]^{-1}$, which implies a Lorentzian spectrum.

We model the signal after heterodyne detection by $dy(t) = s(t,\underline{I})dt + du(t)$, where $s(t,\underline{I})$ is the signal component, in which $\underline{I} = (i_0,i_1,\ldots,i_{N-1})$ is the information-carrying data-sequence, and $u(t)$ is a Wiener process with parameter $N_0/2$. For binary phase-shift-keying, we have $s(t,\underline{I}) = (2P_{IF})^{\frac{1}{2}}\sin[2\pi f_{IF}t+w(t)+\theta+I(t)]$, with the subscript IF denoting "intermediate frequency", and $I(t) = i_n$ for $nT < t < (n+1)T$.

The maximum likelihood estimate $\hat{\underline{I}}$ of the information sequence $\underline{I}$ in terms of the received waveform $\{y(t): 0 \le t \le NT\}$ is the sequence $\underline{I}_j$ of N symbols that maximizes the loglikelihood function

$$\ell(\underline{I}_j) = \int_0^{NT} \hat{s}(t,\underline{I}/y_{0,t},\underline{I}_j)dy(t) - \frac{1}{2}\int_0^{NT} \hat{s}^2(t,\underline{I}/y_{0,t},\underline{I}_j)dt,$$

where $y_{0,t}$ denotes the observed waveform in the interval $[0,t)$, and $\hat{s}(t,I/y_{0,t},\underline{I}_j)$ denotes the causal minimum square-error estimate of $s(t,\underline{I})$ given the observations and assuming the information sequence $\underline{I}_j$. So $\hat{s}(t,\underline{I}/y_{0,t},\underline{I}_j) = E[s(t,\underline{I})/y_{0,t},\underline{I}_j]$.

The minimum mean square-error estimates $\hat{s}$ required above are impractical to implement. Consequently, we introduce an approximation wherein these estimates are replaced by suboptimal estimates s^* so that the information sequence $\underline{I}$ is reproduced as $\underline{I}^*$, where $\underline{I}^*$ is the sequence $\underline{I}_j$ that maximizes

*This work was supported by the National Science Foundation under Grant ECS-8113266.

$$\ell^*(\underline{I}_j) = \int_0^{NT} s^*(t,\underline{I}/y_{0,t},\underline{I}_j)dy(t) - \frac{1}{2}\int_0^{NT} s^{*2}(t,\underline{I}/y_{0,t},\underline{I}_j)dt,$$

where $s^*(t,\underline{I}/y_{0,t},\underline{I}_j) = (2P_{IF})^{\frac{1}{2}}\sin[2\pi f_{IF}t + x^*(t) + I_j(t)]$, in which $x^*(t)$ satisfies the extended Kalman equation

$$dx^*(t) = v^*(t)(2/N_0)(2P_{IF})^{\frac{1}{2}}\cos(2\pi f_{IF}t + x^*(t) + I_j(t))dy(t),$$

and $v^*(t)$ is the solution to the Riccati equation

$$dv^*(t)/dt = \tau_c^{-1} - (2P_{IF}/N_0)v^{*2}(t).$$

The suboptimal estimate s^* is motivated by Van Trees [5] and Snyder [6].

To reduce the complexity of the receiver for long sequences, we use the results in [7] and approximate the memory of the receiver by a few time constants of the extended Kalman equation.

REFERENCES

1. Y. Yamamoto and T. Kimura, "Coherent Optical Fiber Transmission Systems," IEEE Journal of Quantum Electronics, Vol. QE-17, pp. 919-935, June 1981.

2. K. Kikuchi, T. Okoshi, M. Nagamatsu and N. Henmi, "Bit-Error Rate of a PSK Heterodyne Optical Communication System and Its Degradation Due to Spectral Spread of Transmitter and Local Oscillator," Electronic Letters, Vol. 19, No. 11, pp. 417-418, May 1983.

3. V.W.S. Chan, L.L. Jeromin and J.E. Kaufman, "Heterodyne Lasercom System Using GaAs Lasers for ISL Applications," Proc. International Communications Conference, Boston, MA, pp. 1201-1207, June 1983.

4. S. Saito and Y. Yamamoto, "Direct Observation of Lorentzian Lineshape of Semiconductor Lasers and Linewidth Reduction with External Grating Feedback," Electronic Letters, Vol. 17, No. 9, pp. 325-327, April 1981.

5. H.L. Van Trees, Detection, Estimation and Modulation Theory, Vol. II, John Wiley and Sons, New York, 1971.

6. D.L. Snyder, The State-Variable Approach to Continuous Estimation, MIT Press, Cambridge, MA, 1969.

7. R.E. Morley and D.L. Snyder, "Maximum-Likelihood Sequence Estimation for Randomly Dispersive Channels," IEEE Trans. on Communications, Vol. COM-27, No. 6, pp. 833-839, June 1979.

A COMPARISON OF BOUNDS AND SIMULATION RESULTS ON ERROR PROPAGATION IN DECISION FEEDBACK RECEIVERS

John J. O'Reilly
University of Essex, Colchester, England.

A. M. De Oliveira Duarte
University of Aueiro, Portugal

Thomas Beck
University of Karlsruhe, West Germany

ABSTRACT

New bounds on error propagation statistics are examined for specific partial and full response decision feedback receiver structures of importance for high data rate optical fibre communications. The bounds are evaluated and compared with simulation results for these receivers.

INTRODUCTION

A new approach to obtaining bounds on error propagation statistics has recently been developed [1]. In this paper we make use of this approach to study the mean error propagation factor (MEPF) and the mean error burst length (MEBL) associated with using decision feedback structures to decode various partial response signal formats of importance in optical communications. We consider also full response signalling over a channel characterised by a Gaussian response. First order reduced bounds, derived from a reduced state transition diagram relating to the error process and noise independent bounds are considered; the performance of the bounds is assessed by reference to simulation experiments. It is found that the first order reduced bounds are generally tight and that for channels with considerable intersymbol interference the noise independent bounds can also be tight.

UPPER AND LOWER BOUNDS ON ERROR PROPAGATION STATISTICS

Error propagation arises in decision feedback receivers since a noise induced error can result in the feedback signal increasing intersymbol interference rather than reducing it. Given an N-tap feedback filter the effects of a decision error will persist for the next N consecutive sampling times and during this interval further errors may be induced giving rise to error propagation. For an M-ary system there are 2M-1 different possible values for hard decision error amplitudes resulting in $(2M-1)^N$ possible error patterns. The dynamic behaviour of the system can be described by a state transition diagram (STD) but for realistic values of N and M the number of states is inconveniently large.

The complexity of the error propagation problem can be contained by selective elimination of error information [1] resulting in an STD with first order reduction. This enables upper and lower bounds on the statistics of error propagation to be obtained which we refer to as first order bounds.

A further reduction of the STD may be effected by introducing the concept of recovery distance as the number of consecutive correct decisions required to return the system to the error free state. States with the same recovery distance may be merged resulting in a second order reduced STD. Once again, associated worst and best case processes may be identified and used to provide upper and lower bounds.

The performance of some of these new bounds for certain partial response systems of interest for optical communications [2] are compared in Table 1 with results obtained by simulation [3]. Table 2 presents full-response results for a Gaussian optical fibre channel.

CONCLUDING REMARKS

A set of error propagation statistics for certain channels of importance in optical communications have been evaluated using a recently developed bounding technique. Corroborative simulation experiments have been performed.

MEAN ERROR PROPAGATION FACTOR							
PR Class	PR Polynomial	1st Order Bounds Lower	1st Order Bounds Upper	2nd Order Bounds Lower	2nd Order Bounds Upper	2nd Order Noise Independent Bounds (Upper)	Simulation Results
1	$1+z^{-1}$	2	2	2	2	2	2.08
2	$1+2z^{-1}+z^{-2}$	1.2	4	1.03	4	4	3.98
3	$2+z^{-1}-z^{-2}$	1.4	2.2	1.1	3.2	4	1.83
4	$1-z^{-2}$	2	2	2	2	2	2.02
5	$1-2z^{-2}-z^{-4}$	1.2	4	1.03	7.8	16	4.02
8	$2-z^{-2}-z^{-4}$	1.4	2.2	1.1	6.3	16	1.9

Table 1. Error Propagation Bounds and Simulation Results for Partial Response Signals.

	1st Order Upper Bounds	Noise Independent Upper Bounds	Simulation
MEPF	2.25	4	2.05
MEBL	2.4	5	2.1

Table 2. Bounds and Simulation Results for a full-response Optical Fibre System with Gaussian dispersion.

REFERENCES

1. J.J. O'Reilly and A.M. de Oliveira Duarte: 'A new approach for obtaining bounds on error propagation and recovery time for decision feedback receivers', Conference on Information Sciences and Systems, Princeton, March, 1984.
2. A.M. de Oliveira Duarte: 'The application of decision feedback techniques to digital and optical communications', Ph.D. thesis, University of Essex, May 1984.
3. T. Beck: 'Microcomputer based software simulation of a decision feedback equaliser', Internal report, Dept. of Elec. Eng. University of Essex.1984

CODING PERFORMANCE FOR A DIRECT-DETECTION OPTICAL FIBER CHANNEL

MARK A. HERRO
LAIZHAO HU
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

SUMMARY

Previous research on coding for direct-detection optical communications has primarily been focussed on the quantum limited channel, and the effect of thermal receiver noise has been neglected. For uncoded systems, considerable analysis has been performed on models which include both thermal noise and an APD (Avalanche Photon Detector). We introduce a general non-additive Gaussian channel model which includes both quantum and thermal noise effects, and use it to obtain the performance of a RS (Reed-Soloman) coded direct-detection optical system using 2^L-ary PPM (Pulse Position Modulation) and an APD.

In an optical channel, when receiver noise is the major source of signal degradation, a Gaussian approximation to the probability density of the receiver output voltage proves to be quite accurate (see [1], [2], [3]). We consider the following model. In any time slot, the source is either on (pulse) or off (no pulse). In the absence of the pulse, the received output voltage has a Gaussian density function with zero mean and variance σ_0^2 (of course the actual mean will not be zero, but we can transform it to zero without any loss in generality). When a pulse is present, the output has a Gaussian density function with mean S and variance $\sigma_1^2=\sigma_0^2+S\cdot F_M$, where F_M is the excess noise factor of an APD with average gain, M. The work in [1] shows the channel output voltage fits this model.

We examine the performance of rate 1/2 RS coded PPM systems. The symbol size of the RS code is 2^L-1 when 2^L-ary PPM is used. The receiver acts as follows. First a threshold level T is set. At each time slot, if the voltage output exceeds the threshold, "pulse exists" is declared, otherwise "no pulse" is declared. The conceptual situation that gives rise to this non-symmetrical binary channel is shown in Figure 1. Among the 2^L time slots in a word, if only one time slot registers a pulse, the corresponding symbol is passed to the RS decoder. We consider two demodulation options when either no pulse or more then one pulse is detected in a word. In the first method, if more than one time slot contains a pulse, one of these time slots is randomly chosen to generate the symbol. If no pulse is detected in the entire word, we randomly choose a symbol. In the second method we declare an erasure except when exactly one pulse is detected in a word and pass this to the decoder as side information. The RS decoder then acts as an error plus erasure decoder. The threshold, T, will affect the channel transition probabilities ε and δ. In the first demodulation method, the probability of being correct is

$$P_c = (1-\varepsilon)\cdot(1-\delta)^{Q-1} + \frac{1}{Q}\cdot\varepsilon\cdot(1-\delta)^{Q-1} + \sum_{i=1}^{Q-1} \frac{1}{i+1}\cdot\binom{Q-1}{i}\cdot(1-\varepsilon)\cdot\delta^i(1-\delta)^{Q-1-i}$$

where $Q=2^L$. The probability of getting an incorrect symbol is $P_e=1-P_c$.

In the second method, if exactly one pulse is detected, it might be either correct or wrong with probabilities $P_c=(1-\varepsilon)\cdot(1-\delta)^{Q-1}$ and $P_e=(Q-1)\cdot\varepsilon\cdot\delta\cdot(1-\delta)^{Q-2}$. Then the erasure probability is given by $P_r=1-P_c-P_e$.

A RS code of length N and minimum distance D can produce a correct word if the number of input symbol errors N_e and erasures N_r satisfy $2\cdot N_e+N_r<D$. The block error probability is then upper bounded by

$$P_B \leq \sum_{i=0}^{N} \sum_{j=t_i+1}^{N-i} \binom{N}{i} \cdot \binom{N-i}{j} \cdot P_r^{\,i} \cdot P_e^{\,j} \cdot P_c^{\,N-i-j}$$

where i is the erasure index, and j is the error index.

We choose some typical receiver parameters to illustrate the analysis, e.g., M=75, F_M=2 and Z=1500 (see [1]). In many practical systems, the dark current is negligible and thus $\sigma_0^2=(Z/M)^2=400$ and $\sigma_1^2=400+2S$. Optimizing over T, we calculate the relationship between the total error probability and the required number of photons per information bit. The resulting error probability is shown in Figure 2 for L=6, 7 and 8. For example, at L=8, using (255, 127) RS code and 256-ary PPM, S=26 photons are needed if no side information is available. When error plus erasure decoding is used, the requirement is reduced to 22 photons. At a data rate of 10 MBPS, the former requirement corresponds to a received signal power of -73 dBm, if a wavelength of 1 μm is used.

It turns out that the receiver performance is very sensitive to the threshold setting. For a fixed signal strength, when T differs from the optimum value, the error probability increases drastically. However, the major degration in performance when a fixed T is used for different signal levels is that the error probability curve will not be as steep as that in Figure 2. Although the fixed threshold setting technique is sub-optimum in performance, it is less sensitive to signal strength and provides a more robust design.

REFERENCES

[1] S.D. Personic, Optical Fiber Transmission Systems, New York, Plenum Press, 1981.

[2] S.D. Personic, P. Balaban, J.H. Bosin and P.R. Kumar, "A Detailed Comparison of Four Approaches to the Calculation of the Sensitivity of Optical Fiber System Receivers," IEEE Trans. Commun., Vol. COM-25, pp. 541-548, May 1977.

[3] R.M. Gagliardi and G. Prati, "On Gaussian Error Probabilities in Optical Receivers," IEEE Trans. Commun., Vol. COM-28, pp. 1742-1747, September 1980.

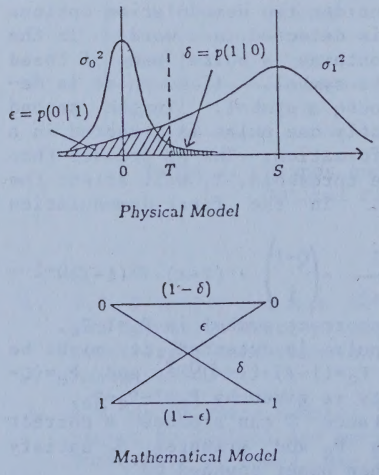


Figure 1. Channel Model

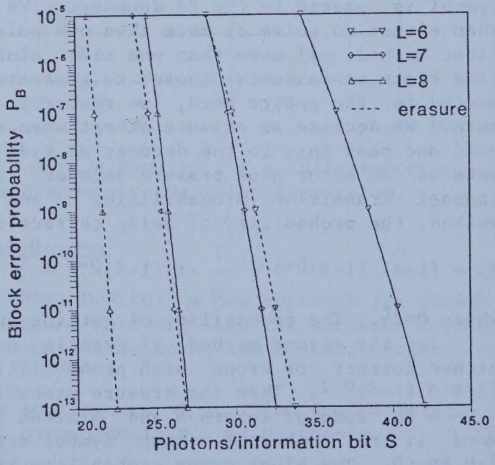


Figure 2. Channel Performance with Optimum T

THE DUAL GHR

D. K. LINDNER
Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

ABSTRACT

An algorithm is presented to transform a linear time invariant system into a canonical form called the Dual Generalized Hessenberg Representation (Dual GHR). The Dual GHR is an extension of the GHR in that it exhibits the input structure as well as the output structure. A geometric interpretation is given and the set of invariants is discussed. It is recognized that this form is essentially the same as the one proposed recently by Kalman [4].

1. INTRODUCTION

It is well known [1-3] that the Generalized Hessenberg Representation (GHR) is a particular representation of a linear time invariant system which explicitly displays a set of subspaces which characterize the output structure. The interaction of this structure with B generates the well known subspaces V^* and R^* and so the invariant zeros can be identified [2,3]. However, a complete description of the interaction of the input and output structure has not been given.

In this paper we study single input single output systems and develop a representation called the Dual GHR, which gives a complete description of the interaction of the input and output structure. In fact, this representation turns out to be essentially the same canonic form as developed by Kalman [4]. In that interesting paper Kalman connects this canonic form to a wide range topics including realization theory, continued fractions, and geometric theory. (See also [5] for an indepth discussion of the realization aspects of this problem.) Thus, the Dual GHR seems to occupy a central role in system theory.

Here we describe how to algorithmically construct the Dual GHR from the state space model. The algorithm contains chained aggregation [1-3] and can be viewed as an extension of that concept. Kalman [4] obtains this form by preforming a certain Euclidean algorithm on the transfer function (thus requiring polynomial division). Gragg and Lindquist [5] show how to obtain the Dual GHR as a realization from the sequence of Markov parameters and corresponding Hankel matrices. To our knowledge, this is the only state space algorithm available.

The geometric analysis of the GHR [2,3] is extended to the Dual GHR. This analysis clarifies how the Dual GHR reflects the input/output structure and reproduces some of Kalman's geometric results [4] in a straight forward manner. The geometric analysis also suggests a set of invariants which differs slightly from those in [4] and [5] but is geometrically more appealing.

In Section 2 we describe the algorithm and give a geometrical intepretation in Section 3. Section 4 comments on the canonical nature of the Dual GHR. Section 5 has the conclusions.

2. THE ALGORITHM

Consider the system

$$\begin{aligned} \dot{x} &= A_1 x + B_1 u, \\ y &= C_1 x, \end{aligned} \tag{2.1}$$

where $x \varepsilon R^n$ and $u, y \varepsilon R^1$. We assume (2.1) is controllable and observable for simplicity.

To simplify the notation, the system (2.1) will be notated by (A_1, B_1, C_1). Let T be a nxn nonsingular matrix. By "preform a transformation on (A_1,B_1,C_1) with T" we mean replace the triple (A_1,B_1,C_1) by $(T^{-1}A_1T, T^{-1}B_1, C_1T)$. If T is chosen such that

$$C_1T = [c_1 \ 0 \ \ldots \ 0], \tag{2.2}$$

then this process will be called one step of chained aggregation. Below, any element in a matrix not explicitly represented is assumed to be zero. Any non-zero elements which are of no interest are denoted by "*".

Given (A_1,B_1,C_1) preform one step of chained aggregation to obtain the system representation

$$\left(\begin{bmatrix} a_1 & C_2 \\ * & * \end{bmatrix}, \begin{bmatrix} b_1 \\ B_r \end{bmatrix}, [c_1 \ \ 0] \right). \tag{2.3}$$

If $b_1 \neq 0$, define the tranformation

$$S_1 = \begin{bmatrix} 1 & 0 \\ B_r b_1^{-1} & I \end{bmatrix} \tag{2.4}$$

and perform a transformation on (2.3).

The result will be

$$\left(\begin{bmatrix} a_1 & * \\ * & * \end{bmatrix}, \begin{bmatrix} b_1 \\ 0 \end{bmatrix}, [c_1 \ \ 0] \right). \tag{2.5}$$

We will call transformations of the type of (2.4) "(Gaussian) elimination of B_r by pivoting on b_1". Finally we scale (2.5) by introducing a scaling transformation N_i which is an identity matrix accept for an α_i in the (i,i) position. With respect to (2.5) we chose α_1 in N_1 such that

$$|c_1 \ \ \alpha_1| = \alpha_1^{-1} b_1 = \gamma_1. \tag{2.6}$$

Hence, we arrive at the system representation

$$\left(\begin{bmatrix} F_1 & C_2 \\ B_2 & A_2 \end{bmatrix}, \begin{bmatrix} G_1 \\ 0 \end{bmatrix}, [H_1 \ \ 0] \right) \tag{2.7}$$

where $F_1=a_1$, $G_1=\gamma_1$ and $H_1=\varepsilon_1\gamma_1$. Here $\varepsilon_1=1$ if $b_1c_1>0$ and $\varepsilon_1=-1$ if $b_1c_1<0$. Set $\sigma_1=1$. Now repeat this process ((2.3)-(2.7)) on the subsystem (A_2,B_2,C_2).

The steps (2.3)-(2.7) embody the essential idea of this algorithm. Note that it inserts zeros into the input and output matrices.

If $b_1 \neq 0$, preform a second step of chained aggregation (with respect to $\bar{C}_2$ in (2.3)) to obtain

$$\left(\begin{bmatrix} a_1 & c_2 & 0 \\ * & * & C_3 \\ * & * & * \end{bmatrix}, \begin{bmatrix} 0 \\ b_2 \\ B_2 \end{bmatrix}, [c_1 \quad 0 \quad 0] \right) . \tag{2.8}$$

If $b_2 \neq 0$, eliminate B_r by pivoting on b_2. The result will be

$$\left(\begin{bmatrix} a_1 & c_2 & 0 \\ * & * & C_3 \\ * & A_r & * \end{bmatrix} \begin{bmatrix} 0 \\ b_2 \\ 0 \end{bmatrix}, [c_1 \quad 0 \quad 0] \right) . \tag{2.9}$$

Now eliminate A_r by pivoting on c_2. This operation yields

$$\left(\left[\begin{array}{ccc} a_1 & c_2 & 0 \\ * & * & C_3 \\ \hline B_3 & 0 & A_3 \end{array}\right] \begin{bmatrix} 0 \\ b_2 \\ \cdots \\ 0 \end{bmatrix} \quad [c_1 \quad 0 \quad 0] \right) . \tag{2.10}$$

Now consider

$$\left(\begin{bmatrix} a_1 & c_2 \\ * & a_2 \end{bmatrix}, \begin{bmatrix} 0 \\ b_2 \end{bmatrix}, [c_1 \quad 0] \right) \tag{2.11}$$

(Compare with the partitioning in (2.10)). We would like to reduce (2.11) to a companion form while retaining the structure in (2.10). This can be done in two steps. First, eliminate a_2 by pivoting on c_2. Then scale the (transformed) system (2.11) by chosing scaling transformations N_1 and N_2 such that

$$\gamma_1 = |c_1 \alpha_1| = \alpha_2^{-1} b_2 \ , \ \alpha_1^{-1} c_2 \alpha_2 = 1 \ . \tag{2.12}$$

The final result is

$$\left(\left[\begin{array}{ccc} a_{11} & 1 & 0 \\ a_{12} & 0 & C_3 \\ \hline B_3 & 0 & A_3 \end{array}\right], \begin{bmatrix} 0 \\ \gamma_1 \\ \cdots \\ 0 \end{bmatrix}, [\varepsilon_1 \gamma_1 \quad 0 \quad 0] \right) . \tag{2.13}$$

Define

$$(F_1,G_1,H_1) = \left(\begin{bmatrix} a_{11} & 1 \\ a_{12} & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \gamma_1 \end{bmatrix}, [\varepsilon_1\gamma_1 \quad 0] \right), \tag{2.14}$$

and set $\sigma_1=2$. Now repeat the entire algorithm with (A_3,B_3,C_3).

If $b_2=0$ then preform chained aggregation until $b_i\neq 0$. Then by a straight foward extension of the algorithm described in (2.9-2.12), the diagonal block in the upper left corner is reduced to a companion form. Note that the scaling also adjusts the constants c_1 and b_i. Set $\sigma_1=i$ and repeat the entire algorithm with $(A_{i+1},B_{i+1},C_{i+1})$.

The description of this algorithm from (2.3) to (2.14) is for one cycle. The algorithm terminates with the r-th cycle when $N[C_r]=0$.

Thus, the system (2.1) is transformed into a representation called the Dual GHR which has the form

$$\begin{bmatrix} F_1 & H_2 & & & & \\ G_2 & F_2 & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & H_r & \\ & & & G_r & F_r \end{bmatrix}, \begin{bmatrix} G_1 \\ \\ \\ \\ \\ \end{bmatrix}, [H_1 \quad], \tag{2.15}$$

where

$$F_k = \begin{bmatrix} a_{k1} & & & \\ \vdots & 1 & & \\ \vdots & & \ddots & \\ \vdots & & & 1 \\ a_{kv_k} & & & \end{bmatrix},$$

$$G_k = \begin{bmatrix} \\ \\ \\ \gamma_k \end{bmatrix}, \quad H_k = \begin{bmatrix} \\ \\ \\ \varepsilon_k\gamma_k \end{bmatrix},$$

$$v_k = \sigma_k - \sigma_{k-1}, \ \sigma_o=0, \ k=1, \ \ldots \ , r. \tag{2.16}$$

If is noticed that, ignoring the other structure, (2.16) is a GHR. Furthermore, if the states are properly re-arranged, the transpose of (2.16) is also in a GHR; hence the name Dual GHR.

3. GEOMETRY

Basic Structure

Consider the following subspaces.

Definition 3.1 [6] The ith unobservability subspace L_i is defined by by

$$L_i \triangleq \bigcap_{j=0}^{i-1} N[CA^j] ,$$

$$L_o \triangleq X. \qquad \square$$

Definition 3.2: [6] The ith reachable subspace R_i is defined by

$$R_i \triangleq \sum_{j=o}^{i-1} A^j B ,$$

$$R_o \triangleq 0. \qquad \square$$

Let e_j be the j-th unit basis vector.

Proposition 3.3.

$$L_i = \sum_{j=i+1}^{n} sp(e_j).$$

$$R_i = R_{\sigma_k} + \sum_{j=\sigma_{k+1}-i+1}^{\sigma_{k+1}} sp(e_j),$$

$$\sigma_k + 1 \leq i \leq \sigma_{k+1}. \qquad \square$$

Note that the subspaces L_i are displayed exactly the same as in the GHR. In addition the subspaces R_i can be easily identified in the Dual GHR.

Decompositions

The Dual GHR identified the subspaced L_i and R_i by specifying a unique basis in which the subspaces L_{σ_i} and R_{σ_i} are orthogonal. In other words, the Dual GHR is dependent on certain topological properties which have not been exploited to date. We can formalize this idea by introducing decompositions.

Definition 3.4 [7] Let X_i, i=1 , ... r be a set of subspaces of R^n (considered as a metric space) such that

1) $X_1 \oplus \ldots \oplus X_r = R^n$,

2) $X_i \perp X_j$, $i \neq j$.

Then we say X_i, i=1, ... r is a decomposition of R^n, or $\{X_i\}$ decomposes R^n. □

The interaction of the subspaces L_i and R_i define a decomposition of the state space in the following way. Define the subspaces K_i by

$$\begin{aligned} R_{\sigma_{i-1}} \oplus K_i \oplus L_{\sigma_i} &= R^n, \\ R_{\sigma_{i-1}} \perp K_i \perp L_{\sigma_i} &, \quad i=1 , \ldots , r. \end{aligned} \tag{3.1}$$

Then the set K_i , i=1 , ... r is a decomposition which is explicitly displayed by the Dual GHR.

In an algebraic setup, the restriction of an operator $A:R^n \to R^n$ to an arbitrary subspace is not well defined. However, in a metric space, we can define a restriction by using (essentially) orthogonal projectors. Let K_i be an orthonormal basis of K_i. The define

$$A|K_i \to K_j = K_j^T A K_i \tag{3.2}$$

For example, given the decomposition (3.1), with reference to (2.15) we have

$$\begin{aligned} A|K_1 \to K_1 &= F_1, \\ A|K_2 \to K_1 &= H_2. \end{aligned} \tag{3.3}$$

c) Zeros

It has been shown [2] that the interaction of L_i and R_1 generates V^*, the supremal (A,B) invariant subspace in N[C]. Using the present notation, we have

Proposition 3.5. [2,3] $V^* = L_{\sigma_1}$ □

The input spaces determine V^* in as much as they determine σ_1. We also have the following characterization of invariant zeros.

Proposition 3.6. Let $R_{\sigma_1} \perp L_{\sigma_1}$ Then the invariant zeros are the eigenvalues of the map

$$A|L_{\sigma_1} \rightarrow L_{\sigma_1}$$

Hence, the invariant zeros are (more or less) explicitly displayed by the Dual GHR. It is satisfying that zeros may be characterized geometrically without recourse to the explicit use of feedback, even if we must go beyond the usual algebraic structure.

The Algorithm

The algorithm of Section 2 is now easily interpreted geometrically. The chained aggregation step identifies L_i [2,3], and the input subspaces R_{σ_k}. Then the subspaces R_{σ_k} and $L_{\sigma_{k+1}}$ are orthogonalized. Finally, the scaling in the subspace K_i is adjusted with the transformations N_i.

4. INVARIANTS

The Dual GHR is a canonic form because the transformation of (2.1) to (2.15) is unique. Define

$$\xi_i(s) = \det (sI-F_i). \tag{4.1}$$

Proposition 4.1. An ordered set of invariants of (2.1) is $\{\xi_i(s), \varepsilon_i \gamma_i , i=1, \ldots , r\}$. □

The polynomials $\xi_i(s)$ are generated by the interaction of the subspaces R_i and L_i. The scalars γ_i, on the other hand, are generated by the relative scaling between K_i and K_j. For instance, we can scale (2.15) such that the nonzero element of G_i equals the nonzero element of H_{i+1}. In that case, we obtain Kalman's invariants [4]. See [5] for another set.

We can interpret this change of scaling, via N_i, as a change in the metric to re-orient the spatial distribution of the invariant subspaces relative to the input and output subspaces. However, the overall input/output properties of the system must remain the same. This property is reflected in the fact that the product $\varepsilon_i \gamma_i \gamma_i$ is invariant to any such scaling. Hence the attractiveness of the invariants of Prop. 4.1.

5. CONCLUSIONS

In this paper we have described a new algorithm to generate the Dual GHR; evidently an important canonic form [4]. We have also given it a geometric interpretation. All of the techniques used here have been applied to multivariable systems [2,3]. The extension of these results to multivariable setting is currently being studied.

6. REFERENCES

1. E. C. Y. Tse, J. Medanic and W. R. Perkins, "Generalized Hessenberg Transformations for Reduced Order Modeling of Large Scale System", Int. J. Control, Vol. 27, 1978, pp. 493-512.

2. D. K. Lindner, W. R. Perkins and J. Medanic, "Chained Aggregation and Three-Control Component Design: A Geometric Analysis", Int. J. Control Vol. 35, 1982, pp. 621-636.

3. D. K. Lindner, "Chained Aggregation and Control System Design: A Geometric Approach", Report R-966 (DC-56), Coordinated Science Laboratory, University of Illinois, Urbana, IL, 1982.

4. R. E. Kalman, "On Partial Realizations, Transfer Functions and Canonical Forms", Acta Polytech. Scand., Vol. MA31, 1979, pp. 9-32.

5. W. Gragg and A. Lindquist, "On the Partial Realization Problem", Linear Algebra and It's Applications, Vol. 50, 1983, pp. 277-319.

6. B. P. Molinari, "A Strong Controllability and Observability in Linear Multivariable Control", IEEE Trans on Automatic Control, Vol. AC-21, 1976, pp. 761-4.

7. D. K. Lindner and W. R. Perkins, "Information Structure and Decomposition of Large Scale Systems", Proc. 22nd Conference on Decision and Control, San Diego, TX, 1983, pp. 515-519.

INVARIANT MANIFOLDS FOR DISCRETIZED SYSTEMS

DAVID L. ELLIOTT
Washington University
St. Louis, Missouri

ABSTRACT: Some simple time-discretizations of continuous systems have the previously unrecognized property that conservative systems lead to conservative discretizations. In the two-dimensional case these methods give topologically correct phase-plane portraits even in the critical (structurally unstable) cases, for linear and nonlinear systems.

Given an autonomous system $dz(t)/dt = f(z(t))$, in real-time computation one cannot use indefinitely small time-steps or excessively high order. If f is nonlinear, chaotic behaviour may result for large time-steps; yet, to my surprise, there are discretization methods for which there exists a computable smooth curve in state space embedding the discrete solution points for each solution.

In digital control and in simulation studies two such methods are usual: the "update" version of Euler's point-slope method, and the (implicit) backward-trapezoidal method. These, at least in the 2-dimensional case, have an unusual virtue: inaccurate though their approximation of $z(t) = (x(t), y(t))$ may be, the computed family of planar trajectories is portrayed accurately (i.e. centers, spirals, saddles, nodes, limit-cycles, ... are correctly located) for moderately large time-steps. The backward-trapezoidal method is to iterate the difference equation

$$z' = z + hf((z+z')/2) \qquad (1)$$

where the prime ' denotes successor, i.e. $z'(t) = z(t+h)$; h is the time-step. It is assumed that $f(0) = 0$ and that (1) can be solved as a power series for z' when h and z are small enough. System (1) is then solvable for z in terms of z' the same way and thus is a diffeomorphism $(x, y) \dashrightarrow (x', y')$. (This can be calculated to any desired order off-line with REDUCE.)

Suppose that the vector field f is conservative in the sense of [Birkhoff]: there exists a scalar function r such that $\mathrm{div}(rf) = 0$. It is my conjecture that if h is smaller than the fastest system time-constant, then bounded solutions of (1) lie on smooth closed curves. Consider the important case $\mathrm{div}\, f = 0$. Then letting fx denote the Jacobian $\partial f/\partial z$, note that $\mathrm{trace}\, f^* = 0$. Letting $J = \partial z'/\partial z$, we have that

$$J = [I - (h/2)f^*]^{-1}\,[I + (h/2)f^*], \qquad (2)$$

J is orthogonal via Cayley's transformation or the new Theorem below, and $\det J = 1$. That is, the transformation from z to z' is locally area-preserving. If f is real-analytic, then for conservative f there exists a formal power series $V(z;h)$ such that

$$V(z'; h) = V(z;h). \qquad (3)$$

This follows from a formal-power-series theorem of [Birkhoff]. The series are computable in REDUCE. They are "asymptotic" usually, and will not converge beyond the polydisc of convergence of the power series for $f(x, y)$; many mathematical questions about convergence remain. However, for n-dimensional LINEAR or BILINEAR systems we have the following new

Theorem: If $dz/dt = A(u)z$ satisfies $dV/dt = 0$, where $V = \langle z, Rz\rangle$ is a quadratic form, then the backward-trapezoidal discretization $z' = z + (h/2)A(u)\,(z + z')$ satisfies $V(z') = V(z)$.

This result provides a long-sought "canonical" discretization for bilinear systems, preserving many Lie-group properties of the continuous system.

The theorem of Birkhoff, and the same questions, apply to the update version of Euler's method, which is explicit and more straightforward to compute:

$$x' = x + hP(y), \qquad y' = y + hQ(x') \tag{4}$$

{ the ' in the right-hand-side of the second equation is not a typo}. In a computer program, say in BASIC, (4) reads x:=x+h*P(y), y:=y+h*q(x) where the x is the just-replaced value. In the "correct" Euler's method, the old value of x is used. It is well known that (since the tangent vector to a circle sticks out of the circle) Euler's method for the simple harmonic oscillator draws spirals instead of the desired circles. Any solution of $x' = x + h\,y$, $y' = y - h\,x'$ lies on an ellipse, that is,

$x^2 + hxy + y^2 = c = x'^2 + hx'y' + y'^2$. In 2n-dimensional space, with $P(y) = Py$, $Q(x') = Qx'$, P and Q symmetric n by n matrices, the ellipsoid $\langle x,Qx\rangle + h\langle Qx,Py\rangle + \langle y,Py\rangle$ is invariant under (4); if P and Q are non-linear but still symmetric, the system (4) is Hamiltonian, and it is known [Bisshopp] that the correction to the Hamiltonian function to compensate for the update discretization is (P(y),Q(x)) correct to order 2 in h. For the second-order system

$$x' = x + hy, \qquad y' = y - h(x')^3 \tag{5}$$

I have used REDUCE 3.0 to compute further correction terms out to degree 7 in h. When (x,y;h) is too far from (0,0;0), chaotic behavior occurs, but near the origin computer solutions lie on smooth ovals.

Formal power series methods can be used to obtain a vector field whose solutions provide a continuous semi-group in which the discrete semigroup of solutions of (4) can be embedded, even for non-conservative systems. The successor operator ' = id + Δ, where Δ is the forward difference operator, has a formal logarithm, a power series in Δ which can be applied to x and to y to obtain the vector field components.

REFERENCES

George D. Birkhoff, "Surface transformations and their dynamic applications," Acta Mathematica Bd.43 (1922) 1-119.

Frederic Bisshopp, Div. Appl. Math., Brown University; private communication.

ZERO ASSIGNMENT WITH MEASUREMENT FEEDBACK

S.O. MAJDI, J.W. HOWZE, AND S.P. BHATTACHARYYA
Department of Electrical Engineering, Texas A&M University, College Station, Texas 77843

ABSTRACT

In this paper we deal with the general linear time invariant system

$$\begin{bmatrix} Y_c(s) \\ Y_m(s) \end{bmatrix} = \begin{bmatrix} G(s) & H(s) \\ M(s) & N(s) \end{bmatrix} \begin{bmatrix} U_c(s) \\ U_r(s) \end{bmatrix}$$

where Y_c, Y_m, U_c and U_r are the control output, measurement output, control input and reference input. For the case where the signals are scalars the problem of assigning a prescribed set of zeroes to the closed loop transfer function relating Y_c and U_r by attaching proper, stabilizing measurement feedback controllers to the system in constructively solved. The set of unassignable zeroes is explicitly characterized. A procedure for finding a controller that assigns the prescribed zeros and makes the system internally stable is described. The results are applicable to the model matching and servomechanism problems.

INTRODUCTION

The asymptotic tracking problem is that of finding a stabilizing, proper controller for a given plant such that the closed-loop system output tracks a given class of reference inputs. In other words, the error transfer function must acquire certain specified zeros.

The solution to the asymptotic tracking and disturbance rejection problem has been given in the state space by Davison [4] and Wonham [9]. Schumacher [7] have a complete solution to the disturbance rejection problem. The equivalent frequency domain solvability conditions have been given by Cheng and Pearson [3], Pernebo [6], and Ohm, Bhattacharyya, and Howze [5]. References [3] and [6] solve the problem completely in the Λ-generalized polynomial domain.

This paper deals with the special case of scalar systems in the s-domain, and the objective is to obtain a simple characterization of the attainable zeros in terms of the original given transfer function. This is the main contribution of the paper in relation to the results of [3] and [6].

PROBLEM FORMULATION

Consider the system represented by the transfer function matrix

$$\begin{bmatrix} y_c(s) \\ y_m(s) \end{bmatrix} = \begin{bmatrix} G(s) & H(s) \\ M(s) & N(s) \end{bmatrix} \begin{bmatrix} u_c(s) \\ u_r(s) \end{bmatrix} \qquad (1)$$

where $G(s) = \frac{n_G(s)}{d(s)}$, $H(s) = \frac{n_H(s)}{d(s)}$, $M(s) = \frac{n_M(s)}{d(s)}$, and $N(s) = \frac{n_N(s)}{d(s)}$ are

proper rational transfer functions with $n_M(s)$ and d(s) relatively prime, and $u_c(s)$, $u_r(s)$, $y_c(s)$, and $y_m(s)$ are the transforms of the control input, the exogenous input, the output to be controlled, and the measured (available) output, respectively. In the following, the variable s is dropped for convenience.

Assume that the system is stabilizable from u_c and detectable from y_m. A test for stabilizability and detectability of the system which is based directly on (1) appears in Bhattacharyya and Howze [2].

Define n to be a polynomial such that

$$n_G n_N - n_M n_H = nd. \qquad (2)$$

The existence of n is guaranteed by stabilizability and detectability [2].

MAIN RESULT

In the following we characterize the set of all attainable zeros.

Let $Z^+_{GN} := \{s \in \mathbb{C}^+ : n_G(s)\, n_M(s) = 0\}$,

$Z^+_{HO} := \{s \in \mathbb{C}^+ : n_H(s) = n(s) = 0\}$,

$Z_\zeta := \{s \in \mathbb{C}^+ : \zeta(s) = 0\}$, and

$N_T := Z^+_{GN} / (Z^+_{GN} \cap Z^+_{HO})$.

THEOREM 1.

There exists a proper stabilizing controller, $u_c(s) = C(s)\, y_m(s)$, that assigns the zeros of $\zeta(s)$ as the zeros of the transfer function relating $y_c(s)$ and $u_r(s)$ in the controlled system, if and only if

$$N_T \cap Z_\zeta = \phi .$$

REFERENCES

[1] B.D.O. Anderson and R.W. Scott, "Output Feedback Stabilization - Solution by Algebraic Geometry Methods." Proceedings of the IEEE, Vol. 65, No. 6, June 1977, pp. 849-861.

[2] S.P. Bhattacharyya and J.W. Howze, "Transfer Function Conditions for Stability." To appear IEEE T-AC, March 1984.

[3] L. Chen and J.B. Pearson, "Synthesis of Linear Multivariable Regulators." IEEE T-AC, Vol. AC-26, No. 1, Feb. 1981, pp. 194-202.

[4] E.J. Davison, "The Robust Control of a Servomechanism Problem for Linear Time - Invariant Multivariable Systems." IEEE T-AC, Vol. AC-21, 1976, pp. 25-34.

[5] D.Y. Ohn, S.P. Bhattacharyya and J.W. Howze, "Transfer Function Condition for (C,A,B) Pairs." To appear IEEE T-AC, April 1984.

[6] L. Pernebo, "An Algebraic Theory for the Design of Controllers for Linear Multivariable Systems - Parts I and II." IEEE T-AC, Vol. AC-26, No. 1, Feb. 1981, pp. 171-194.

[7] J.M. Schumacher, "Regulator Synthesis Using (C,A,B,) Pairs." IEEE T-AC, Vol. AC-27, No. 6, Dec. 1982, pp. 1211-1221.

[8] W.A. Wolovich, Linear Multivariable Systems, Springer-Verlag, 1974.

[9] W.M. Wonham, Linear Multivariable Control: A Geometric Approach, Springer-Verlag, 1979.

OPTIMIZATION OF DECENTRALIZED EIGENVALUE ASSIGNMENT

DALE R. SEBOK
School of Electrical Engineering
Purdue University
West Lafayette,IN 47907

RAYMOND DECARLO
School of Electrical Engineering
Purdue University
West Lafayette,IN 47907

ABSTRACT

This paper gives an algorithm for eigenvalue assignment using optimal feedback. The algorithm transforms a nonoptimal feedback into an optimal one by means of an iteration formula. The formula also allows for small movements of the eigenvalues of the system. The formula is derived from the first order variational behavior of the eigenvalue-eigenvector equations.

INTRODUCTION

There exist today many algorithms which will assign a spectrum to a system [5,6]. Most of them, however, fail to make use of the non-uniqueness inherent in the eigenvalue assignment problem. Despite the contributions of [7,8] the feedback gains produced by the algorithms presently available are often unrealistically large. The purpose of this paper is to give an algorithm which will optimize the result of any valid eigenvalue assignment algorithm. More specifically we seek an algorithm which will take an initial feedback matrix which assigns a user specified spectrum and modify it into one which minimizes a quadratic cost functional over all feedbacks which assign the given spectrum. The algorithm presented in this paper allows for the use of output and decentralized feedback. In fact, due to the structure of the algorithm , it is particularly suited for use in conjunction with the continuation algorithm for eigenvalue assignment by decentralized constant output feedback[1-3].

THEORY

The iteration formula is base on the following state model

$$\dot{x} = [F + BK_d C]x \tag{1}$$

where B is a block-diagonal input matrix , C is a block-diagonal output matrix, and K_d represent the decentralized feedback and is also block diagonal. This model corresponds to the composite state model of a system described by a CCM(Component Connection Model) where the blocks inside the matrices correspond to the components of the system. For more information on the CCM and the development of equation 1 see [1-3]. Note that the above model reduces to the standard state model when the system has only one component (one block in each matrix).

The goal of this paper is to give an algorithm which will allow for the determination of a feedback matrix ,K_d, of the appropriate block structure which has the minimum cost and is contained in the set of feedback matrices which assign the user specified spectrum $\{\Lambda\}$. For the purposes of this paper we will assume that a feedback, K_d, is known such that $\sigma(F+BK_d C) = \{\Lambda\}$ where $\sigma(\cdot)$ denotes "the spectrum of". The algorithm described in [1-3] can be used to find this K_d. From this initial condition the iteration formula 2 is repeatly applied. It can be shown that by using an appropriate γ repeated application of 2 will converge to the minimum K_d and assign the specified spectrum, ie. the resulting K_d minimizes $J(\mathcal{K}) = \mathcal{K}^t Q\mathcal{K} + R^t\mathcal{K} + \xi$ under the constraint that $\sigma(F+BK_d C) = \{\Lambda\}$ where Q is symmetric positive definite, R is an arbitrary column vector, and ξ is an arbitrary constant, and $\mathcal{K} = \mathrm{vec}(K_{d_1}, \cdots K_{d_N})$ where K_{d_i} correspond to the local feedback matrices. For the definition of the vec() operator and the tensor product ($\otimes$) see [4]. The iteration formula which will converge to the minimizing $\mathcal{K}$ is

$$\mathcal{K}_{new} = (I - \gamma V(V^t QV)^{-1}V^t Q)(\mathcal{K}_{old} + \zeta\delta\mathcal{K}_{LS}) - 0.5\gamma V(V^t QV)^{-1}V^t R \tag{2}$$

where V is a matrix of maximal rank satisfying $P(\mathcal{K})V = 0$, $V^t V = I$, $P(\mathcal{K}) = E(\mathcal{K})[C^t \otimes B]$, $E(\mathcal{K}) = [(e_1^t \otimes w_1^t)^t \cdots (e_n^t \otimes w_n^t)^t]^t$ and where e_i and w_i are the right and left eigenvectors associated with equation 1 satisfying $\langle e_i, w_j \rangle = \delta_{ij}$. $\delta\mathcal{K}_{LS} = P^+(\mathcal{K})\,\delta\Lambda$ represents the correction required to zero out the eigenvalue error given in the vector $\delta\Lambda$. The magnitude of this

correcting term is govern by the constant ζ. The constant γ controls the magnitude of the optimizing term. From a theoretical standpoint if γ is made infinitesimally small ζ may be set to zero since $\delta\Lambda$ will be zero. From a practical point of view due to finite iteration steps and finite arithmetic there will be a finite error in the eigenvalues assigned and hence ζ must be include in the algorithm. In a practical implementation of the algorithm γ and ζ are dynamically assign such that the following conditions hold.

1. The spectrum is invariant.
2. The cost of the feedback matrix will decrease at any value which is not minimal.

Equation 2 can be developed by writting the variational equation relating the variations in the feedback to the variations of the spectrum and eigenvectors. In this variational equation the variation of the eigenvectors appears as second order terms and therefore can be neglected for sufficiently small variations in the feedback. Note that in equation 2 the variation in $\mathcal{K}$ can be made arbitrarily small by adjusting γ and ζ. After neglecting the eigenvector terms and setting the variations of the spectrum to zero (ζ=0) an equation is found which yields all small movements allowed in the feedback which have the same spectrum. From this set an iteration formula is determined by restricting consideration to feedback variations which decrease the system cost. Finally the resulting iteration formula is shown to converge only to local minimums of the cost function and hence by construction satisfies the two properties given above. These two properties along with the convex nature of the cost function insure convergence of the algorithm to the desired optimal feedback matrix.

RESULTS

$I_p(\cdot)$ was applied to a two component system described in example 5 in [1] with $Q = I, R=0$, and ξ =0. The system components have the following number of states,inputs and outputs respectively, 6,4,6 and 5,3,5. To this system the following spectrum was to be assigned, $\{\Lambda\}$ = {-66.3,-6+j,-6-j,-27.6,-6,-100,-71.1,-8+1.5j,-8-1.5j,-26.4,-8}. The system represents a system containing two interconnected synchronous generators. Dynamic compensation was added to component 1 to satisfies the even-odd condition of [1-3]. For further details of this CCM and example see [1].

The program in [1] was used to find a non-optimal value of K_d. The resulting cost for this K_d was 19,073. The iteration formula outlined in this paper was then applied till K_d converged to the optimal K_d. The optimal K_d had a cost of 1,738 which represents an order of magnitude improvement in the cost function.

ACKNOWLEDGMENT

We acknowledge the early work and input of Serge Lefebvre and Stephen Richter which helped give rise to the results of this paper.

REFERENCES

[1] T. Harris and R.A. DeCarlo, "Numerical and Theoretical Investigations into a Continuous Approach to Decentralized Eigenvalue Assignment," Purdue, School of Electrical Engineering, TR-EE 82-11, May 1982.

[2] Steve Richter and R. DeCarlo, "A Homotopy Method for Eigenvalue Assignment Using Decentralized State Feedback," *IEEE Trans. Automat. Contr.*, Vol. AC-29, No. 2, February 1984

[3] S. Lefebvre, S.Richter, and R.DeCarlo, "A Continuation Algorithm for Eigenvalue Assignment by Decentralized Constant Output Feedback," accepted for publication, *International Journal of Control*.

[4] J.W. Brewer, "Kronecker Products and Matrix Calculus in System Theory," IEEE Trans. on Circuits and Systems, Vol. CAS-25, No. 9, September 1978, pp. 772-781.

[5] W.M. Wonham, *Linear Multivariable Control: A Geometric Approach*, New York: Springer Verlag, 1979.

[6] G.S. Miminis and C.C Paige, "An Algorithm for Pole Assignment of Time Invariant Linear Systems," *International Journal of Control* ,Vol. 35, No. 2, 1982, pp.341-354.

[7] H. Kimura, "On Pole Assignment by Output Feedback," *International Journal of Control* ,Vol. 28,No. 1.

[8] R.V. Patel, "Computational Algorithms for Pole Assignment in Linear Multivariable Systems," *Proceedings of the Second IFAC Symposium on Computer Aided Design of Multivariable Technological Systems*, West Lafayette, IN, September 1982.

POLE ASSIGNMENT BY USING DYNAMIC OUTPUT FEEDBACKS

SHOU-YUAN ZHANG

Department of Electrical Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794

SUMMARY

In this paper, we present the results of pole assignment by dynamic output feedback with transfer function approach. In particular, we consider the pole assignment for arbitrary distinct real, repeated real and complex conjugate poles by an m order compensator. We first study the single-input multi-output systems, then extend the result to multi-input multi-output systems.

Consider a strictly proper single-input multi-output system $G(s)=N(s)d^{-1}(s)$ with q outputs, where N(s) and d(s) are right coprime and degree of d(s) is n.

Lemma 1: If N(s) and d(s) are right coprime, the rank of the m+1 order resultant of $\{N(s),d(s)\}$ is m+k+1, where $k=k_0+k_1+\ldots+k_m$, and k_i is the number of observerbility indices of G(s) larger than or equal to i.

Lemma 2: Let A be a full rank VxU constant matrix, U > V. Then for almost all full rank UxV constant matrix, say B, the matrix AB has a full rank V.

Lemma 3: Consider a monic polynomial with degree n, say,

$$a(s)=s^n+a_{n-1}s^{n-1}+\ldots+a_0$$

If there exists a set of coefficients, a_0, a_1,..., a_{n-1}, such that the following equation is met:

$$[a_0 \quad a_1 \quad \ldots \quad a_{n-1} \quad 1]\, Z = 0$$

where Z is an arbitrary (n+1)xm matrix of rank m, $m \leq n$, then m zeros of a(s) can be arbitrarily assigned.

With these lemmas, we prove the following theorem:

Theorem 1: Consider a strictly proper 1-input q-output system $G(s)=N(s)d^{-1}(s)$, where N(s) and d(s) are right coprime and degree of d(s) is n. If an m order compensator is employed, m+k poles of the feedback system can be almost arbitrarily assigned.

We prove this theorem by using the following equation:

$$[K_0 \quad h_0 \quad K_1 \quad h_1 \quad \ldots \quad K_m \quad h_m]S_{m+1}Z = [f_1 \quad f_2 \ldots \quad f_{n+m}]Z = 0$$

Since S_{m+1} has rank m+k+1 and the last d-row of S_{m+1} is always independent of the previous rows, the other rows in S_{m+1} will have rank m+k. We choose Z according to the pole assignment requirement, then for almost all systems with the same set of observerbility indices as that in Theorem 1, there exists a set of coefficients K_0, K_1,...,K_m, and h_0, h_1,..., h_m, such that the above equation will be satisfied. Hence, we also have the following Corollary.

Corollary 1: For almost all 1-input q-output systems with the same set of observerbility indices as that of the system in Theory 1, there exists an m order compensator such that m+k poles of the feedback system can be arbitrarily assigned.

Theorem 1 and Corollary 1 provide a tight criterion for

pole assignment of single-input multi-output systems by m order compensators. The role played by the observerbility indices of the systems is clarified.

For multi-input multi-output systems, we know that the following lemma exists:

Lemma 4: Consider a qxp strictly proper cyclic system G(s). For almost all lxq and pxl real constant vectors T_1 and T_2, we have

$$c.p.G(s)=c.p.T_1G(s)=c.p.G(s)T_2$$

where c.p. denotes the characteristic polynomial.

It is known that almost all constant output feedback can transform a noncyclic system into a cyclic system. By using these results, any strictly proper system can be transformed to a single-input multi-output system without lossing the structure information. To obtain a tight criterion for pole assignment, we also need this lemma:

Lemma 5: The observerbility indices of the multi-input multi-output system are identical with the single-input multi-output system transformed by using Lemma 4.

Theorem 2: Consider a strictly proper qxp system $G(s)=D^{-1}(s)N(s$ with left coprime D(s) and N(s). If an m order compensator is employed, m+k poles of the feedback system can be almost arbitrarily assigned.

Similar to the single-input multi-output case, we have the following Corollary:

Corollary 2: For almost all qxp system with the same set of observerbility indices as that of the system in Theorem 2, there exists an m order compensator such that m+k poles can be arbitrarily assigned.

We note that the result of Theorem 2 can be improved due to the fact that the control channels have been increased. In fact, when we use a pxl constant vector T_2 to transform a multi-input system into a single-input system, almost all pxl vectors can be used. Hence, we have freedom to choose p-1 parameters. This forms a parametrization of the closed loop characteristic polynomial, which could be used to assign more poles or to stabilize the feedback system.

Theorem 3: Consider the design problem of Theorem 2. In addition to assign m+k closed loop poles almost arbitrarily, we have a parametrization of p-1 arbitrarily assignable parameters for the characteristic polynomial of the feedback system.

REFERENCES:

1. H.Kimura, "Pole Assignment by Gain Output Feedback," IEEE Trans. Automat. Contr., Vol. AC-20, 1975.
2. E.J.Davison and S.H.Wang, "On Pole Assignment in Linear Multi-variable Systems Using Output Feedback,"IEEE Trans. Automat. Contr., Vol. AC-20, 1975.
3. A.S.Morse, W.A.Wolovich and B.D.O.Anderson,"Generic Pole Assignment: Preliminary Results," IEEE Trans. Automat. Contr. Vol. AC-28, 1983.
4. T.E.Djaferis, "Generic Pole Assignment Using Dynamic Output Feedback," Int. J. Contr., Vol. 37, 1983.
5. P.G.Antsaklis and W.A.Wolovich, "Arbitrary Pole Placement Using Linear Feedback Compensators," Int. J. Contr. Vol.25, 1977.

ON A POLE ASSIGNMENT PROPERTY OF KLEINMAN'S STABILIZATION RULE

H. SELBUZ, E. YAZ
Department of Applied Mathematics
The Institute for Basic Sciences
P.O.Box 74 Gebze-Kocaeli, Turkey

ABSTRACT

The effect of the stabilizing feedback gain of Kleinman [1] on single-input constant discrete-time systems is brought out. The pole assignment to origin is investigated changing a certain design parameter. In this way, the results of the previous works are generalized and a more complete picture is obtained for this type of controller affecting a certain plant.

MAIN RESULT

Given a linear time-invariant system defined in discrete-time of the following form

$$x_{i+1}=Ax_i+Bu_i \quad , \; x_o \text{ given} \tag{1}$$

where $x_i \in \mathbb{R}^n$, $u_i \in \mathbb{R}^m$ and (A,B) is completely controllable, Kleinman [1] proved that the constant feedback control

$$u_i = -R^{-1}B^T A^{(r-1)^T} W_r^{-1} A^r x_i \tag{2}$$

stabilizes system (1) if A is nonsingular and r is so chosen that $r \geq n+1$. Here $R^T=R>0$ and W_r is given by

$$W_r \triangleq Q_r Q_r^T \quad , \; Q_r \triangleq [B, AB \ldots, A^{r-1}B] \tag{3}$$

If A is singular, the same control law, under the same conditions, renders system (1) asymptotically stable and this was shown by Kwon and Pearson [2] for m=1 and m=n, and by Kamen and Khargonekar [3] for multiinput systems.

In this work, we investigate what happens when a control of a type similar to (2) is applied to system (1) for m=1 and r<n+1.

Now, consider the single-input system

$$x_{i+1} = Ax_i+bu_i \quad , \; x_o \text{ given} \tag{4}$$

where $u_i \in \mathbb{R}$ and A is singular in general. We use the control law

$$u_i = -K_r x_i \tag{5}$$

where

$$K_r = b^T A^{(r-1)^T} W_r^{\dagger} A^r \quad , \quad r=1,2,\ldots,n \tag{6}$$

in which W_r has been defined in (3), and $W_r^{\dagger}$ denotes the Moore-Penrose pseudo-inverse of W_r. The main result of this paper is summarized in the following theorem:

Theorem: When the feedback control (5) is applied to system (4) the resulting closed loop system has at least r number of poles which are assigned to origin. If r is taken to be equal to the state dimension, all the poles of the closed loop system are assigned to origin meaning that K_n defined by (6) is a state deadbeat control gain.

Proof: For the proof of this result, see Selbuz and Yaz [5].

This theorem may even hold for multiinput systems but this remains to be shown. When A is nonsingular, K_n is identical to the control gain which was obtained via the receding horizon control approach by Yaz and Selbuz [4]. Since for r=n, the receding horizon approach give a deadbeat control, the

same was to be expected of the Kleinman's controller. In this way, a complete picture of Kleinman's controller is obtained for single-input systems. When r is taken as r<n, at least r poles of the closed loop system are assigned to zero. When r=n, all the poles are assigned to zero and one obtain a deadbeat control. If r>n, it has allready been shown that all the closed loop poles are assigned into the unit circle as has been mentioned. In case r<n, one can not predict the places of the remaining n-r number of poles which are not assigned to zero. For example, one might think whether these poles are closer to the unit circle than their unstable open loop counterparts or that they remain the same. The example given below will illustrate us that one cannot say much about the places of these remaining poles..

Example: Consider the following system

$$A=\begin{bmatrix} 5 & 1 \\ 0 & 2 \end{bmatrix} \quad , \quad b=\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Obviously the system is controllable, If we make a design for r=1, the closed loop eigenvalues are 0, and 3. The nonzero pole, as one may notice, is not equal to any of the open loop poles and, furthermore, is not assigned into the the stable region. The design for r=2=n assigns both poles of the closed loop system to zero.

REFERENCES

[1] D.L.Kleinman, "Stabilizing a discrete, constant, linear system with application to iterative methods for solving the Riccati equation", IEEE Trans.Automat.Contr., AC-19, pp.252-254, 1974.

[2] W.H.Kwon and A.E.Pearson, "On the stabilization of discrete constant linear system", IEEE Trans.Automat.Contr., Vol.AC-20, pp.800-801, 1975.

[3] E.W.Kamen and P.P.Khargonekar, "On the control of linear systems whose coefficients are functions of parameters", IEEE Trans.Automat.Contr., Vol.AC-29, pp.25-33, 1984.

[4] E.Yaz and H.Selbuz, "A note on the receding-horizon control method", Int.J.Contr., Vol.39, pp.853-855, 1984.

[5] H.Selbuz and E.Yaz, "Pole placement consideration of Kleinman's rule", (submitted for publication).

STATE FEEDBACK DESIGN FOR ROBUST MULTIVRIABLE SYSTEMS

N.K. Sinha
McMaster University
Hamilton, Canada

P.Rozsa
Budapest Technical University
Budapest, Hungary

SUMMARY

It is well known that the eigenvalues of a controllable system can be placed at specified locations by providing suitable state feedback. The solution for the feedback gains is unique only for the SISO case. For the multivariable case, there are many possible solutions for the feedback gain matrix which give the same eigenvalues for the closed-loop system. This flexibilty can be exploited to improve the design of the closed-loop system [1-5]. In particualr, it has been shown by Gilbert in a recent paper [6] that the norm sensitvity of the eigenvalues to variations in parameters is minimized if the eigenvectors of the resulting system matrix form a mutually orthogonal set.

In practice, it will seldom be possible to satisfy this condition of orthogonality completely. Hence, it is more realistic to make them as nearly orthogonal as possible. A scheme for accomplishing this will be presented in this paper.

Consider a completely controllable linear system described by

$$\dot{x} = A\,x + B\,u \tag{1}$$

With state feedback

$$u = K\,x \tag{2}$$

we have

$$\dot{x} = (A + BK)\,x = A_f\,x \tag{3}$$

For a multivariable system the matrix K can be selected in such a way as to locate the eigenvalues of A_f as desired. The eigenvectors of A_f may also be assigned by the choice of K (with certain restrictions).

Following Kailath [3], let the transfer function matrix be

$$G(s) = C\,(sI - A)^{-1}\,B = N(s)\,D^{-1}(s) \tag{4}$$

where D(s) is a column reduced polynomial matrix. It can be written as

$$D(s) = D_{hc}\,S(s) + D_{lc}\,\Psi(s) \tag{5}$$

where

D_{hc} = highest column degree coefficient matrix of D(s)

$S(s) = \mathrm{diag}\,[\,s^{k_i}, \quad i = 1, 2, \ldots, m]$

$D_{lc}\,\Psi(s)$ accounts for the remaining lower column degree terms of D(s).

D_{lc} is a matrix of coefficients whereas,

$$\Psi'(s) = \begin{bmatrix} s^{k_1 - 1} \ldots s\ 1 & 0 & 0 & 0 \\ 0 & s^{k_2 - 1} \ldots s\ 1 & 0 & 0 \\ & & & \\ 0 & 0 & 0 & s^{k_m - 1} \ldots s\ 1 \end{bmatrix}$$

All of these matrices can be written by inspection if the system is in the controllor canonical form [3].

The closed-loop system transfer function is obtained as

$$G_f(s) = N(s)\ D_f^{-1}(s) \qquad (6)$$

where

$$D_f(s) = D(s) + K\ \Psi(s) \qquad (7)$$

The eigenvectors of the closed-loop system can be assigned as desired provided that they are chosen from the range space of $\Psi(\lambda_i)$. It may be noted that λ_i are the eigenvalues of A_f. Due to this restriction, it will not usually be possible to make the eigenvectors mutually orthogonal. Therefore, one can only try to make them as nearly orthogonal as possible.

In this paper, a scheme for doing this is proposed, but due to space limitations, it cannot be described here in detail. The basic idea is to minimize the sum of the squares of the scalar products of the various eigenvectors.

REFERENCES

1. B.C. Moore, "On the flexibilty offered by state feedback in multivariable systems beyond closed-loop eigenvalue assignment," IEEE Trans. Automat. Contr., vol. AC-21, pp. 689-692, 1976.

2. G. Verghese and T. Kailath, "Fixing the state-feedback gain by Choice of Closed-loop Eigensystem," Proceedings 1977 IEEE Conference on Decision and Control, New Orleans, pp. 1245-1248, 1977.

3. T. Kailath, "Linear Systems," Prentice-Hall, Inc., 1980.

4. V. Gourishankar and K. Ramar, "Pole assignment with minimum eigenvalue sensitivity to parameter variations," Int. J. Contr., vol.23, pp. 493-504, 1976.

5. K. Ramar and V. Gourishankar, "Utilization of the design freedom of pole assignment feedback controllers of unrestricted rank," Int. J. Contr., vol. 24, pp. 423-430, 1976.

6. E.G. Gilbert, "Conditons for minimizing the norm sensitivity of characteristic roots," IEEE Trans. Automat. Contr., vol. AC-29, pp. 658-661, 1984.

PARALLEL COMPUTATION OF NORMALIZED FOURIER DESCRIPTORS

KIRK D. SMITH and LEAH H. JAMIESON
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

Normalized Fourier descriptors provide an effective but computationally intensive method for performing object identification and tracking. Parallel algorithms to compute normalized Fourier descriptors are presented. The task includes sub-algorithms for conversion of chain code inputs to X-Y coordinates, filtering, resampling, Fourier transform, and normalization. MIMD and SIMD formulations are considered. The algorithms are analyzed with respect to computational complexity and communications requirements. For typical problem sizes and appropriate choice of machine size P, speedups of O(P) are achieved.

1. INTRODUCTION

Image processing algorithms are growing more complex as research is conducted. Performance demands are also increasing steadily. The major factor fueling these advances is increased speed of computer hardware. Practically, speed of computing hardware has some limitations. In the future, gains in speed may not be as dramatic. Even today, certain tasks cannot be performed because of computational bottlenecks and real-time requirements.

Clearly, a solution to these problems lies in the replication of available computing hardware. A challenging part of this field lies in the development of parallel algorithms for varied image processing tasks. In this paper, an implementation of an image processing algorithm is presented. It is representative of a wide class of image processing algorithms since it is composed of several sub-algorithms, each with different characteristics.

2. ALGORITHM OVERVIEW

Given the contour of an object in a two dimensional plane as input, a series of frequency domain coefficients which describes the image is computed. These are the Fourier descriptors, which are further processed in a normalization procedure so that they can be compared to a library of these descriptors. Output of the algorithm is the identification of the object as well as a reasonable estimate of its orientation in space [9]. The algorithm has been proven effective in identifying and tracking aircraft in flight [10].

Input to the program consists of the chain code representation of the contour of an image in a two dimensional plane. In practice, chain code inputs typically contain from 64 to 2048 points. This chain code input is then converted to X-Y coordinates of the image. After optional smoothing, the image is resampled at equally spaced intervals on the contour. Then a complex Fourier transform is performed on the resampled points. This produces a Fourier descriptor (FD).

The second logical division of the algorithm normalizes this descriptor. The goal is to scale and orient the contour by rule such that the FD from an unknown contour will always normalize to the correct library representation. Different normalizations have been proposed. Wallace's algorithm [9] is investigated here.

This material is based on work supported by the U.S. Army Research Office under Contract DAAG29-82-K-0101.

The normalization is accomplished as follows: The N most significant complex coefficients of the FD (N is typically 32), are denoted as $A(-N/2 + 1)$ through $A(N/2)$. This frequency domain representation of the contour is normalized by removing information relating to the relative position of the contour, its size, its starting point, and its orientation. This is accomplished by three steps:

Step 1 Set $A(0) = 0$.
This removes all "DC" positional information.

Step 2 Divide A(i) by $|A(1)|$, $-N/2 + 1 < i < N/2$
By definition, A(1) will be the largest coefficient, so this normalizes the size of the image such that $A(i) \leq 1$, for all i.

Step 3 Multiply the A(i)'s by $e^{j[(i-k)u + (1-i)v]/(k-1)}$
k is the coefficient with second largest magnitude (after A(1))
u and v are the phases of A(1) and A(k) respectively
This simultaneous application of the rotation and starting point shift operations finds one of the normalizations satisfying $u = v = 0$. This places a major axis of the contour along the X-axis.

If $k = 2$, this normalization is unique. Otherwise the phase and starting point of the normalization must be shifted to account for the $|k - 1| - 1$ other possible normalizations. Then the correct normalization must be chosen based on some other criteria. The criterion examined here chooses the correct normalization as the one which maximizes

$$\sum_{i=-N/2+1}^{N/2} \text{Re}[A(i)]\,|\,\text{Re}[A(i)]|$$

A parallel implementation for the complete FD algorithm will be presented. The algorithm is divided into distinct tasks, each of which is examined individually. To achieve further parallelism, the tasks could be pipelined to increase throughput for real-time applications.

3. MACHINE MODELS

Two models of asynchronous parallel processing and one model of synchronous parallel processing will be used in the algorithms. The asynchronous models will be MIMD (Multiple Instruction Stream - Multiple Data Stream) machines; the synchronous model will be an SIMD (Single Instruction Stream - Multiple Data Stream) machine [2].

The organization assumed for an SIMD machine will be a set of P processing elements (PEs), each a processor with its own memory; a control unit; and an interconnection network. The control unit broadcasts instructions to all PEs, and each active PE executes the instruction on the data in its own memory. The interconnection network allows data to be transferred among the PEs. Examples of this model are MPP (Massively Parallel Processor) [1] and Siegel's PArtitionable SIMD/MIMD (PASM) system [5]. An MIMD machine will be assumed to consist of P processors, M memories and an interconnection network. Each processor can execute an independent instruction stream. In the Shared Memory MIMD model, the interconnection network is used to allow all processors access to all of memory. Examples of this model include the NYU Ultracomputer [4] and C.mmp [3]. In the Private Memory MIMD model, there is no global store, each processor has a local memory ($M = P$), and the interconnection network provides communications among separate processors. An example of this is PASM [5].

Since communication is a critical part of parallel algorithms, the types of communications needed by each of the algorithms will be analyzed. This will be a function of the way in which the data is distributed among the processors/memories. However, for a given data allocation, the precise communications requirements can be obtained for SIMD and Private Memory MIMD systems. These will be expressed in terms of a few common interconnection functions. In SIMD mode, the transfer will occur simultaneously for all active processors p, $0 \leq p < P$; in a Private Memory

MIMD machine, the transfer will be a request from one PE to obtain data from another PE's memory. (In SIMD algorithms it more common to consider transferring data *from* a given PE *to* another PE. In MIMD algorithms, it may be more natural to consider a data access as a transfer of data *from* a non-local memory *to* a given PE. Since the interconnection functions we are using are symmetric, this distinction will not matter.) The interconnection functions needed will be:

(1) the class of *shift* functions, where shift $\pm d$ transfers data from PE p to PE $(p \pm d) \bmod P$.

(2) the class of *cube* functions, where if $r = \log_2 P$ and $p_{r-1} \cdots p_i \cdots p_0$ is the binary representation of p, $0 \leq p < P$, then

$$cube_i(p) = p_{r-1} \cdots \overline{p_i} \cdots p_0$$

where $\overline{p_i}$ is the complement of p_i.

4. DECOMPOSITION INTO PARALLEL ALGORITHMS

In this section, parallel algorithms are described for each of the subtasks required for generating normalized Fourier descriptors. The algorithms are for conversion from chain code to X-Y coordinates, filtering, resampling, Fourier transform calculation, and FD normalization.

4.1 Input Conversion

It will be assumed that the contour of the image is entered in chain code representation. Figure 1a shows a typical representation for an 8 nearest neighbor chain code. The location of point p_i is dependent upon the points p_0 through p_{i-1}. The horizontal and vertical segments (0,2,4,6) have length 1; the diagonal segments (1,3,5,7) have length $\sqrt{2}$.

An example 25-point contour with its chain code representation is given in Figure 1b. Chain code inputs of practical use contain from a few hundred to a few thousand points. This is a variable number which depends on the relative size, shape, and perspective of the object being identified. The number of chain code inputs will be assumed to be C.

Chain code input is inherently serial since each input is merely an offset from the previous input. Two parallel algorithms for this normally serial task will be described. Initially assume that $P = \sqrt{C}$. Thus, the C inputs can be divided among P PEs and each PE will be responsible for $\sqrt{C}$ chain code inputs. This can be illustrated by forming the input logically into a two dimensional array. The array of input points will be denoted as CCIn(0..C−1). Figure 2 shows the division of the contour into segments and arranges each segment into a line of the array. We can then define parallel operations in which each PE acts on a row or column of this array.

The first parallel algorithm uses the Shared Memory MIMD model. Initially each row of the input is processed by a separate PE. This is equivalent to dividing the contour into P contiguous segments, with each PE responsible for one segment. Each PE can then assume that it has the "first" segment of the contour and assign the coordinates (0,0) to the first point. It can then compute the X-Y coordinates of the rest of its points starting from this reference. Given $\sqrt{C}$ chain code inputs in each segment, each PE assumes the first point, then generates X-Y coordinates for $\sqrt{C}$ additional points. Thus, the last point generated in PE p corresponds to the first point for PE p+1 (the point previously assumed to be (0,0)). X-Y coordinates for all the input points have now been generated, however each row of the square (each segment of the contour) has a different origin in the X-Y plane. Now a correction step is employed. Denote the X-Y coordinates of input point i as XY(i), $0 \leq i < C$. Since the origin is arbitrary, set it at the point XY(0), that is $XY(0) \equiv (0,0)$. Then the previous step correctly computed the coordinates of XY(0)

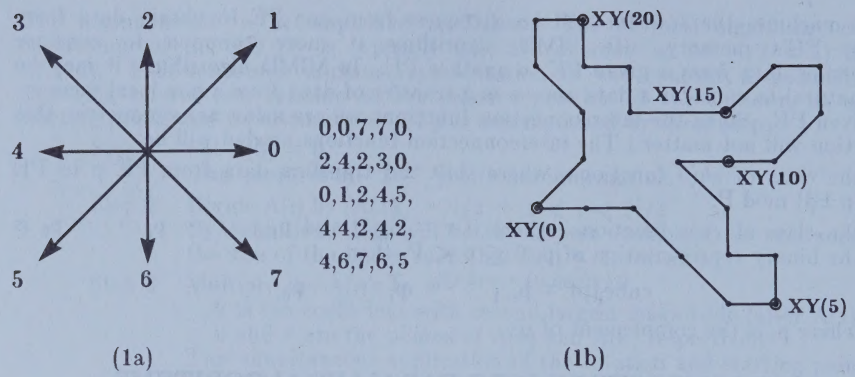


Fig. 1a. Typical chain code representation
Fig. 1b. Example 25-point chain code and contour

CCIn(0)	CCIn(1)	CCIn(2)	···	CCIn(P−1)
CCIn(P)	CCIn(P+1)			·
CCIn(2P)				·
·				·
·				·
·				·
CCIn((P−1)P)	·	·	·	CCIn(P^2−1)

Fig. 2. Division of chain code points

through XY(P). To correct the coordinates of XY(P) through XY(2P) in the second segment, we must add to each of these the coordinates of XY(P) computed in the first segment. Subject to memory access constraints, these P+1 corrections can be done concurrently. Then to correct points XY(2P) through XY(3P) in the third segment we must add the (newly corrected) XY(2P) from the second segment. All of the segment S can be corrected in parallel, however, segment S must be corrected before segment S+1, for $1 \leq S < P-1$. This correction must be done in order, for each row of the square.

This algorithm can easily be generalized to any number of input points by assigning $\lceil C/P \rceil$ consecutive points to each of the first P − 1 processors, and $C-(P-1)\lceil C/P \rceil$ points to the last processor. Some efficiency will be lost if all processors do not contain the same number of points.

In order to estimate the amount of computation performed, some assumptions about the number and types of statements will be made. Initially, synchronization overhead and memory conflicts will not be considered. The basic operations performed in the parallel algorithm are the conversion from chain code input to X-Y coordinates based on an arbitrary origin and the correction of the X-Y coordinates to the correct origin. Assume that the functions CtoX() and CtoY() convert one chain code input to the proper increment in the X or Y direction. This can be done with a simple case statement or a conversion table in which each of the 8 possible chain code inputs maps to the appropriate X and Y increments. Then the coordinates for

XY(i+1) can be obtained from XY(i) and the i-th chain code input by the pair of statements:

$$X(i+1) \leftarrow X(i) + CtoX(CCIn(i))$$

$$Y(i+1) \leftarrow Y(i) + CtoY(CCIn(i))$$

This can be considered to be 2 additions and assignments using real arithmetic or one addition and assignment using complex arithmetic. The serial algorithm consists of C calls to the conversion functions, C complex additions, and C complex assignments. The parallel approach executes in the time for $\sqrt{C}$ calls to the conversion functions, $\sqrt{C}$ assignments, and $\sqrt{C}-1$ complex additions for the initial conversion, plus $\sqrt{C}-1$ complex additions and assignments for the correction. Assuming the dominant operation is the additions, the speedup on computations is approximately

$$S \simeq \frac{C}{2\sqrt{C}-1} \simeq \frac{C}{2\sqrt{C}} = \frac{1}{2}\sqrt{C}\,.$$

For $P = \sqrt{C}$, $S \simeq P/2$.

Consider the memory references required in the above algorithm. If the data is viewed as a matrix with P data points on a side, each processor operates on a row and then a column of that matrix. In a parallel system with global memory, the store is typically divided into several memory units. Optimum efficiency comes about when each processor is accessing a different memory unit during a given memory cycle, since each memory unit can deliver only one word per memory cycle. An obvious way to distribute the data is to put each segment of the contour (row of the matrix) in a separate memory unit. During the first half of the conversion, each processor acts on a row, so the memory system operates with ideal efficiency. During the second half of the conversion, every processor acts on the same row simultaneously. This creates a large bottleneck at the memory unit containing that row. Kuck discusses this problem in [Kuc77] and suggests skewed storage techniques that eliminate these bottlenecks at the cost of more complex address computations in array accesses. There is an overhead involved in every array access, thus reducing the speedup.

In the SIMD or Private Memory MIMD model, it is assumed that accesses to the local memory can occur without contention from other processors. Consider rewriting the algorithm to use only local memories. The initial step of the algorithm is unchanged: each PE obtains X-Y coordinates for one segment of the contour, assuming an arbitrary origin. Then recursive doubling is done to produce correction values for all the segments of the contour at once. Recursive doubling is a method of computing accumulated sums across processors [8]. To show this in a program segment, assume a call to rec_dbl(val) uses the value *val* and takes care of all the communications to perform the recursive doubling. If val(i) refers to the value of *val* in PE i, then for all PEs p, $0 \leq p < P$, rec_dbl(val(p)) will return the partial sum:

$$\text{rec_dbl(val(p))} = \sum_{i=0}^{p} \text{val(i)}$$

An example of recursive doubling is shown in Figure 3.

For the Private Memory algorithm, the restriction that $P = \sqrt{C}$ is relaxed. The only assumption made is that there are D input points in each processor's local memory in the array CCIn(0..D−1). This algorithm stores a segment of the contour in each PE's local memory. Initially, each PE computes coordinates based on the assumption that its first point is at the origin (0,0). Knowing the relative coordinates of the last point in each segment, the absolute coordinates of the beginning of each segment can be computed as follows. The correction for PE 1 is given by the coordinates of the last point in PE 0; the correction for PE 2 is given by the sum of the coordinates of the last points in PEs 0 and 1; in general, the correction for PE p is the sum of the coordinates of the last points in PEs 0 through p − 1. Recursive doubling is used to compute all the needed sums simultaneously.

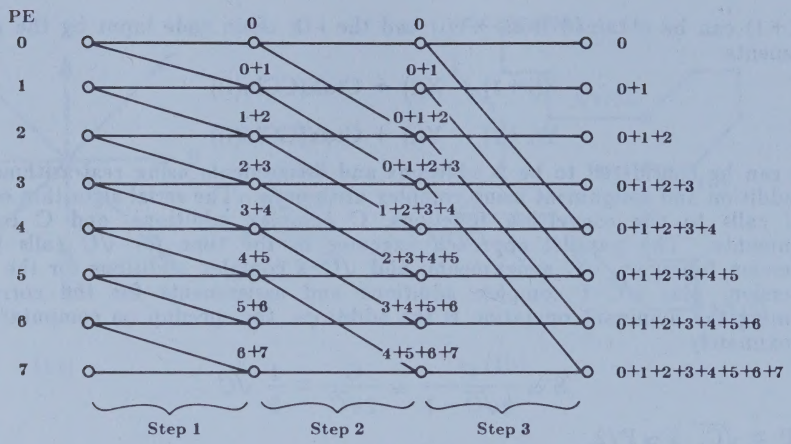


Fig. 3. Recursive doubling example for 8 PEs

Once each PE has the absolute coordinates for its first point, it can correct the rest of its segment locally. This step will be done concurrently in all PEs. The algorithm is given in Figure 4.

```
/* Local variable definitions */
P               /* Number of processing elements */
D               /* Number of data points in each processor */
CCIn(0..D-1)    /* Input chain code for one contour segment */
X(0..D)         /* X coordinates for this contour segment */
Y(0..D)         /* Y coordinates for this contour segment */
sumx            /* Partial sum of all X coordinates */
sumy            /* Partial sum of all Y coordinates */

X(0)←Y(0)←0
sumx←sumy←0

/* Compute X-Y coordinates for all points */
FOR i←0 THROUGH D-1 DO
BEGIN
        X(i+1)←X(i)+CtoX(CCIn(i))
        Y(i+1)←Y(i)+CtoY(CCIn(i))
END

/* Compute correction factors in parallel */
sumx←rec_dbl(X(D)) /* log2P transfer steps */
sumx←sumx-X(D) /* Only consider offset from previous segments */
sumy←rec_dbl(Y(D)) /* log2P transfer steps */
sumy←sumy-Y(D) /* Only consider offset from previous segments */

/* Correct each segment locally */
FOR i←1 THROUGH D-1 DO
BEGIN
        X(i)←X(i)+sumx
        Y(i)←Y(i)+sumy
END
```

Fig. 4. Input conversion algorithm for a Private Memory MIMD machine

Here the number of input points is $C = P*D$. The overall computational complexity is proportional to $D+\log_2 P$. If the assumption is kept that $C = P^2$, then $D = P = \sqrt{C}$, and the complexity is $\sqrt{C}+\log_2\sqrt{C}$, compared with a complexity proportional to C for the serial algorithm. The operations in the first part of the algorithm (i.e., the local chain code to X-Y coordinate conversions) are the same as those used in the serial algorithm, but are performed in C/P steps instead of C. The remainder of the parallel algorithm is all overhead. The recursive doubling requires $\log_2 P$ complex additions and assignments. It also requires $\log_2 P$ points of synchronization. The local correction step takes time proportional to C/P; however, each operation is simply a complex addition, which takes less time than the original chain code to X-Y coordinate conversion step. The time is therefore dominated by the original conversion step. An O(P) speedup is expected; accounting for the extra steps, an actual speedup of P/2 is conservative.

Summarizing, two algorithms for the input conversion have been presented. Both methods are fairly regular and could be done on an SIMD machine. The first method is well suited for a Shared Memory MIMD machine, and the second method works well with either MIMD machine model. The first method effectively uses broadcasts (by placing values in memory), while the second method uses shift $+2^i$ functions, $0 \leq i < \log_2 P$, for the recursive doubling. Consider representing the complexity of the algorithm as being proportional to $\alpha C/P + \log_2 P$. The actual choice of P will in general be made based on speed constraints of the application and the range of values of C. We would like to estimate P_{max}, the largest "reasonable" value for P. As an arbitrary measure, if we say that we want the cost of the computation to dominate (i.e., $\alpha C/P > \log_2 P$) and let $\alpha = 2$, then for small contours ($C = 64$), $P_{max} = 16$, and for large contours ($C = 2048$), $P_{max} = 256$.

4.2 Filtering

The filtering of the image is an optional step to remove some of the quantization noise. Typically this is a smoothing operation, in which each point is replaced by the (possibly weighted) average of itself plus W neighboring points. This can be done easily in parallel by giving each processor a section of the contour. Given a filtering window width W, each processor will need to access $(W/2)-1$ points from each adjoining section. This could be accomplished by at most W transfer steps. If a memory system is used where accesses to adjacent memories are allowed, it is important that "wrap-around" can occur. That is, PE P−1 should be a "neighbor" to PE 0. For more discussion of the filtering problem in general, see [7].

Overall, in this portion of the algorithm speedups on the order of P can be expected for small values of W. For large W, the number of accesses to data in adjacent processors may be significant. Then, properties of the parallel system will have a greater effect on the total processing time. These properties include methods of memory accesses and interconnection between processors/memory.

In the filtering step, only shift ± 1 communication is needed. The number of usable PEs is related to the number of points per PE and the width W. Speed constraints may dictate how many points must be filtered by each PE. Speedup can be increased by increasing P. On the other hand, for large W, the relative effect of the transfers can be reduced by decreasing P and thus increasing C/P. Since filtering is a regular operation, it could be done easily on SIMD as well as MIMD machines.

4.3 Resampling

The input outline needs to be resampled since the Fourier descriptor algorithm requires equal distances between input samples. From chain code input, the diagonal segments are longer by a factor of $\sqrt{2}$.

The basic approach to resampling is to compute the length of the entire C-point contour, then resample it to R evenly spaced points. The length within each PE can

be computed with a speedup of P and the partial and total sums across the PEs can be computed in $\log_2 P$ steps using recursive doubling. After the total length has been obtained, the contour is divided into P groups of points such that all groups have equal length. Each PE will compute the resampled points for its own group. In the conversion and filtering stages, each PE held the same number of points. Here the PEs hold equal length groups of points, and the number of original points between two groups may differ by as much as $\sqrt{2}$. Once the boundary locations between groups have been determined, contour points may have to be moved between adjacent PEs to achieve the division into equal length groups. Once the appropriate points are collected into a PE, it can compute the resampled points for its own group. Since each PE's group has the same length, each PE will compute the same number of resampled points.

During this resampling, each processor operates primarily on local data. The only need for non-local data occurs at the ends of the contour segments. The amount of non-local data required depends on the resampling technique employed. Since most data is local, memory access is not a problem. During the recursive doubling, the interconnection network will be used. Thus, any architecture in which the communications facilities can easily support nearest neighbor (shift ± 1) and recursive doubling (shift $+2^i$) transfers should run this algorithm well.

Although SIMD machines can be used for resampling [11], MIMD execution is more suitable here because of the possible irregularities in the distances between the original samples. Either MIMD model should perform well. Again, P_{max} is chosen so that the number of points in each PE is large enough so that the amount of work is significant compared to the parallel overhead. The range of P_{max} as a function of C will be approximately the same as for the input conversion algorithm.

4.4 Fourier Transform

The FD is obtained by computing the first 32 points of the DFT on the R-point resampled contour. Here an FFT algorithm utilizes the PEs well. Since the number of contour points may be as large as 2048, but only 32 frequency domain coefficients are required for the FD, it may seem that a DFT, computing only the 32 coefficients needed, could provide similar speedups. Unfortunately, the DFT suffers from the need to broadcast all R points to all PEs, and does not approach the low computational cost of the FFT for the range of R of interest.

Using the parallel FFT algorithms in [6], a radix-2 R-point FFT can be computed in P processors (P a power of 2) in $\frac{R}{2P}\log_2 R$ complex multiplication steps, $\frac{R}{P}\log_2 R$ complex addition steps, and $\frac{R}{2P}\log_2 P$ transfer steps. If, for example, $P = 32$, then each PE will hold R/32 input samples, and the execution is dominated by $\frac{R}{64}\log_2 R$ complex multiplications and $\frac{R}{32}\log_2 R$ complex additions. In addition, $\frac{5}{64}R$ transfer steps are needed. These transfer steps represent the overhead of parallel execution and could account for execution overhead near the time required for the multiplication and addition steps. Even so, for $R >> P$, the speedups are no worse than P/2, thus the asymptotic speedup for this portion of the algorithm is O(P). In order to accomplish these gains, a communications facility is required to transfer the data at each point of synchronization. Because of the high degree of synchronization required, the FFT is best suited for SIMD rather than MIMD implementation. The data transfers are cube_i functions, $0 \leq i < \log_2 P$, and are done frequently. An MIMD machine of either type would be slowed by the large amount of communication and synchronization.

For an R-point radix-2 point FFT, as many as R/2 PEs could be used. However, it will be practical to use the same number of PEs as were used in the previous algorithm (resampling), so that data reallocation is not needed.

4.5 Descriptor Normalization

The remaining step normalizes the coefficients by rule so that they can be compared to a library of contour coefficients. The number of coefficients is N which is typically 32. Suppose $P = N = 32$. To normalize the coefficients, A(0) is set to 0 and all values are scaled by $|A(1)|$. This requires one broadcast and one parallel division. To find which coefficient is largest, the magnitudes can be computed in parallel, then the comparison can be performed in $\log_2 P$ (=5) transfers and comparisons using recursive doubling. The speedup would be on the order of only $P/\log_2 P = 6.2$ for this small section. Then depending on parameters of A(0) and A(k), the starting point and origin are shifted appropriately. This is done once if $k = 2$, otherwise it is done $|k - 1|$ times. Speedups can be estimated from the operations involved in shifting the origin or starting points. Either of these can be computed easily by multiplying each coefficient by a complex factor. This factor is the same across all processors for the origin adjustment and it is computed individually for the starting point adjustment. No communication or synchronization is needed, so any MIMD system should handle these well. The speedups then will be $S \simeq P = 32$ for these shifting operations.

When more than one of these normalizations are done, the "correct" normalization is computed as the one with the maximum sum

$$\sum_{i=-N/2+1}^{N/2} \mathrm{Re}[A(i)]\,|\mathrm{Re}[A(i)]|$$

These terms can be computed in parallel with optimal speedup. The sum can then be formed in $\log_2 N$ steps and compared on a single processor to each of the other normalizations.

Instead of dealing with a few hundred or a few thousand data points of varying number, this procedure deals with only N ($N \simeq 32$) data points. Thus, care must be observed in estimating speedups since synchronization overhead may be a significant factor in execution speed. For this part of the processing a SIMD system may be more efficient. MIMD systems would need to have efficient synchronization mechanisms to perform well. Overall, however, a somewhat lesser speedup in this section will not significantly affect the execution time, since the number of data items has dropped from several hundred to 32, and the complexity of the operations performed in this step is not high. Since each PE could contain as few as one point each, P_{max} could be 32.

5. CONCLUSIONS

The use of a parallel machine could speed up the normalized Fourier descriptor algorithm significantly. For practical contours, the number of PEs that can be effectively used is approximately 16 or 32. The types of transfers required are the cube_i, shift ± 1, and shift $+2^i$ functions, for $0 \leq i < \log_2 P$. Some sub-algorithms are better suited to MIMD architectures, others to SIMD architectures. Together, the collection of algorithms that comprise the FD task demonstrates a variety of techniques in parallel processing and shows that substantial speedups can be achieved using parallelism.

ACKNOWLEDGMENTS

The authors thank O. R. Mitchell for his help in understanding the essence of the Normalized Fourier Descriptor algorithm.

REFERENCES

[1] K. E. Batcher, "MPP -- A Massively Parallel Processor," *1979 Int'l Conf. Parallel Processing*, Aug. 1979 p. 249.

[2] M. J. Flynn, "Very High-Speed Computing Systems," *Proceedings of the IEEE*, Vol. 54, 1966, pp. 1901-1909.

[3] S. H. Fuller, R. J. Swan, W. A. Wulf, "The Instrumentation of C.mmp, a Multi-(mini)processor," *COMPCON 73 Digest of Papers*, Mar. 1973, pp. 173-176.

[4] A. Gottlieb, et al., "The NYU Ultracomputer -- Designing an MIMD Shared Memory Parallel Computer," *IEEE Trans. Comp.*, Vol. C-32, Feb. 1983, pp. 175-189.

[5] H. J. Siegel, et al., "PASM: A Partitionable SIMD/MIMD System for Image Processing and Pattern Recognition," *IEEE Trans. Comp.*, Vol. C-30, Dec. 1981, pp. 934-947.

[6] L. J. Siegel, P. T. Mueller, Jr., H. J. Siegel, "FFT Algorithms for SIMD Machines," *Seventh Annual Allerton Conf. on Communication, Control, and Computing*, Oct. 1970, pp. 1006-1014.

[7] L. J. Siegel, H. J. Siegel, A. E. Feather, "Parallel Processing Approaches to Image Correlation," *IEEE Trans. Comp.*, Vol. C-31, Mar. 1982, pp. 208-218.

[8] H. S. Stone, "Parallel Computers," in *Introduction to Computer Architecture*, H. S. Stone, ed., Science Research Associates, Chicago, 1980, pp. 372-403.

[9] T. P. Wallace and O. R. Mitchell, "Local and Global Shape Description of Two and Three-Dimensional Objects," School Elec. Eng., Purdue Univ., West Lafayette, IN, Tech. Rep. TR-EE 79-43, Sept. 1979.

[10] T. P. Wallace and O. R. Mitchell, "Analysis of Three-Dimensional Movement Using Fourier Descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, Nov. 1980, pp. 583-588.

[11] M. R. Warpenburg and L. J. Siegel, "SIMD Image Resampling," *IEEE Trans. Comp.*, Vol. C-31, Oct. 1982, pp. 934-942.

MARKOV MODEL FOR IMAGE SEGMENTATION [1]

NAI-KUAN HUANG [2]
Department of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455

ABSTRACT

First order statistics is used to detect texture edge in 1-D profiles. Textures are modeled as unobservable states of a Markov chain, and the gray levels observed are modeled as probabilistic outputs at the states. Since the state transition probabilities and the output distributions are assumed unknown, this method is useful even though no prior texture information of the picture is given. State distribution at each pixel can be effectively computed after maximum-likelihood estimates of these unknown parameters are obtained via an iterative algorithm. A step-like change in the state distribution indicates an edge. The amount of computation required for the processing is in proportion to the degree of spatial resolution times the degree of intensity resolution of the image. A complete processing of a 512x512 image with 8 gray levels needs approximately thirty million multiplications and additions per pass. Usually ten passes on each line give recognizable boundaries of the textured regions.

1. INTRODUCTION

In this paper, we introduce a new way to detect texture edges using first order statistics of the gray levels. It is true that two entirely distinct textures can have the same first order statistics, but very often, in images of natural scenes or microscopic pictures, differently textured regions have different first statistics. It is much easier to begin the segmentation process using this information first, and, if necessary, leave the remaining problem to the next stage of segmentation employing second order statistics.

The approach we take is, to treat a number of 1-D profiles separately, and do a line-tracing to connect the edges into smooth boundaries. Before this fine segmentation, we can crudely determine some regions where texture boundaries are suspected to exist by testing some statistical hypothesis over a number of small areas, as described in [9]. With the aid of the coarse segmentation, we can process at each time a portion of a 1-D profile that contains exactly one texture edge.

In general there is no problem to process a 1-D profile that consists of a fixed number of textures, as will be clear in the next section. Since we assume no prior knowledge of the textural content in the image, we do not know the number of textures in any region. The easiest way to get around this problem is to do a pre-processing as mentioned above.

---

[1]This work is supported by the Army Research Office under Contract Nos. DAAG29-79-C-0054 and DAAG29-82-K-0142.

[2]This author was with the Communication Sciences Institute, Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90085-0272.

2. IMAGE MODEL

We view the detection of texture boundary in a digitized picture as a state estimation problem. This model considers the gray levels as probabilistic output of a (2-state) Markov chain. In this paper, we treat the case when each state of the Markov chain is a random source with unknown output distribution. The Markov source case is treated in [8].

A diagram of this model is in the figure, where:

s_1, s_2 are two different discrete sources, each representing a one-dimensional texture.

e(t), t=1,2,3,... is a random switch which assumes its value at either the state s_1 or s_2. Crossing over boundary from one texture to another will produce a change of the state of e(t).

n(t), t=1,2,3,... is an additive memoryless noise due to some uncontrollable factors.

y(t), t=1,2,3,... is the observed data to be processed. The problem of one-dimensional texture edge detection translates into that of estimating the state of e(t) at each instant t, given a collection of y(t)'s.

In the next section, we will show how to estimate the position of the random switch at every instant t. This model can be used to study bursty channels [7], seismic structures [11], logic circuitry reliability [5], and language models [1]. When the sources are Markov, rather than independently distributed, more states are required [8].

3. MATHEMATICS

Let S be a finite set of size s, called the state space. Let O be a finite set of size r, called the observation space, or alphabet. There should be no confusion to call the states 1,2,...,s, and the members of the alphabet 1,2,3,...,r. Members of S will be denoted by x, and of O by y. We will also denote random variables with values in S by X, and random variables with values in O by Y.

Let $A=(A_{ij})$ be the transition matrix of a Markov chain with state space S, i.e.,

$$\sum_{j=1}^{s} A_{ij}=1, \text{ for } i=1,2,\ldots,s, \text{ and } A_{ij}\geq 0 \text{ for } i,j=1,2,\ldots,s.$$

Let $B=(B_{ij}^{k})$ be a tensor of output probabilities, where B_{ij}^{k} is the probability of observing k given an (ij) transition occurs, and satisfies:

$$\sum_{k=1}^{r} B_{ij}^{k}=1, \text{ for } i,j=1,2,\ldots,s, \text{ and } B_{ij}^{k}\geq 0 \text{ for } i,j=1,2,\ldots,s \text{ and } k=1,2,\ldots,r.$$

Let $P=[p_j]$, j=1,2,...,s, be an initial distribution vector, where p_j is the probability that the initial state of the Markov chain is j.

We define a random variable N, which stands for, given two numbers x and x', the count of their occurrence along a random sequence, then

$$\sum_{t=0}^{N-1} P_\theta(X_t=x,\ X_{t+1}=x' \mid [Y=y]_1^N)=E\{N(x,x') \mid [Y=y]_1^N\}. \tag{1}$$

An effective way to compute the *a posteriori* probabilities is the theme of the next two sections. That, in turn, leads to a recursive algorithm to estimate the unknown parameters A and B.

For convenience, we define:

$$C_{ij}^k=A_{ij}B_{ij}^k$$

which is the joint probability of a transition from i to j, and observing k. Note that

$$\sum_k C_{ij}^k=A_{ij}$$

so we can always recover A,B from the C's, and they must satisfy

$$\sum_{j,k} C_{ij}^k=1.$$

3.1 Forward-Backward Recursions

Define

$$\alpha_t(x)=P_\theta(\{[Y=y]_1^t\} \text{and} \{X_t=x\}),$$

which is the joint probability of observing $[Y=y]_1^t$ and the state x where the path ends, and

$$\beta_t(x)=P_\theta(\{[Y=y]_{t+1}^N\} \mid \{X_t=x\})$$

which is the conditional probability of observing $[Y=y]_{t+2}^N$ given that the path starts at state x.

Using probability partition and factorization techniques, we obtain

$$\alpha_t(x_t)=\sum_{x_{t-1}} \alpha_{t-1}(x_{t-1}) C_{x_{t-1}x_t}^{y_t} \tag{2}$$

which is a forward recursive relation, with initial condition $\alpha_o(x)=p_x$.

Notice that

$$\sum_{x_N} \alpha_N(x_N)=P_\theta([Y=y]_1^N) \tag{3}$$

which is exactly the quantity we want to maximize.

Similarly we obtain

$$\beta_t(x_t)=\sum_{x_{t+1}} C^{y_{t+1}}_{x_t x_{t+1}} \beta_{t+1}(x_{t+1}) \tag{4}$$

which is a backward recursive relation with initial condition $\beta_N(x_N)=1$.

It is clear from the definition of $\alpha_t(x)$ and $\beta_t(x)$ that:

$$\text{Prob}(\mathbf{X}_t=x \mid [\mathbf{Y}=y]_1^N)=\alpha_t(x)\beta_t(x)/\text{Prob}([\mathbf{Y}=y]_1^N),$$

$$\text{for } x=1,2,\ldots,s, \text{ and } t=1,2,\ldots,N, \text{ where } \text{Prob}([\mathbf{Y}=y]_1^N)=\sum_{x_N}\alpha_N(x_N). \tag{5}$$

3.2 Reestimation

The probability of state transition xx' occurs at time t given the observation $[y]_1^N$ is:

$$\text{Prob}(\mathbf{X}_t=x, \mathbf{X}_{t+1}=x' \mid [\mathbf{Y}=y]_1^N) = \frac{\text{Prob}(\mathbf{X}_t=x, \mathbf{X}_{t+1}=x', [\mathbf{Y}=y]_1^N)}{\text{Prob}([\mathbf{Y}=y]_1^N)}$$

and after a few manipulation, we can express this in terms of the α,β and C's so that

$$\text{Prob}(\mathbf{X}_t=x, \mathbf{X}_{t+1}=x' \mid [\mathbf{Y}=y]_1^N)=\frac{\alpha_t(x)C^{y_{t+1}}_{x_t x_{t+1}}\beta_{t+1}(x')}{\sum_{x''}\alpha_N(x'')}$$

A set of new parameters $(C^{v}_{xx'})^*$ can be obtained by normalizing the sum of these expected counts, according to equation 1, with constraint $\sum_{x',y}(C^{y}_{xx'})^*=1$; i.e., we can use

$$(C^{y}_{xx'})^* \sim C^{y}_{xx'}\sum_{t=0}^{N-1}\alpha_t(x)\beta_{t+1}(x') \tag{6}$$

with the righthand size properly normalized according to the above constraint. This is the *key equation* of the reestimation algorithm. It is proved [2, 3, 4, 10] that in most reasonable cases, the reestimated C's converges monotonely to the maximum likelihood estimate of the unknown parameters.

4. EDGE DETECTION

Since we treat each texture as a state in a two-state hidden Markov chain, those pixels in the first texture are said to be in the first state, and transit back to the first state, until at the boundary point, the transition will be from the first state to the second state. And from then on those pixels in the second texture are said to be in the second state, and will remain in the second state till the end of the line. A typical transition matrix of this Markov chain is:

$$A = \begin{vmatrix} 1-p & p \\ 0 & 1 \end{vmatrix},$$

where p is determined by the length of pixels in the first texture. In practice, we do not know this length in advance, and will initially use a subjective estimate of p. The reestimation algorithm described in the previous chapter will refine it in the process.

The gray tone of each pixel is treated as probabilistic output at the associated state. The discrete density functions may be obtained from a comprehensive frequency count along the available lines or line segments that contain only a single texture. But again, we do not assume this knowledge in our automatic texture boundary detection. These parameters are initiated by a weighted frequency count. All that is assumed is that there is a boundary point somewhere along the line. Experiment shows that the weighting functions employed can be either linear or step, or the combination of both, as long as they are monotone decreasing. Different set of B's due to different initial weighting will merge into the same path of convergence within a few reestimations.

Computation involved in an NxN image with r gray levels is $9rN^2$ additions and multiplications. With N equals 512 and r equals 8, this is well in the range of current main frame computer for near-real-time processing without any parallelism.

Synthetic textures generated in [6] are used for experiment. Edge between textures can be nearly perfectly detected (within two pixels) in all cases, provided each piece of texture is at least 16 pixels long. This method fails to accurately detect edges between two separately equalized textures, as expected. A many-state image model reported in [8] has successfully solved the more difficult problem.

REFERENCES

[1] L.R. Bahl, F. Jelinek, and R.L. Mercer.
A Maximum Likelihood Approach to Continuous Speech Recognition.
IEEE Transactions on Pattern Analysis and Machine Intelligence
PAMI-5(No. 2):179-190, March 1983.

[2] L. Baum and T. Petrie.
Statistical Inference for Probabilistic Functions of Finite State Markov Chains.
Annals of Math. Stat. 37, 1966.

[3] L.E. Baum and J. A. Eagon.
An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology.
Amer. Math. Soc. Bull. 73:360-363, May 1967.

[4] L.E. Baum, T. Petrie, G. Soules and N. Weiss.
A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chain.
Ann. Math. Stat. 41:164-171, 1970.

[5] T.H. Chen.
Logic Hazard Analysis.
Chungqing University Mathematics, 1980.

[6] D. Garber.
Computational Models for Texture Analysis and Texture Synthesis.
PhD thesis, University of Southern California, May 1981.

[7] E.N. Gilbert.
Capacity of a Burst-Noise Channel.
Bell System Technical Journal:1253-1265, 1960.

[8] N.K. Huang, L.R. Welch, R.A. Scholtz.
Markov Texture Model for Automatic Segmentation.
Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[9] S. Kulback.
Information Theory and Statistics.
John Wiley and Sons, 1959.

[10] L.A. Liporace.
Maximum Likelihood Estimation for Multivariate Observations of Markov Sources.
IEEE Transactions on Information Theory IT-28(No. 5):729-734, 1982.

[11] J.M. Mendel.
Minimum-Variance Deconvolution.
IEEE Tran. on Geo. Sci. and Rem. Sen.:161-171, 1981.

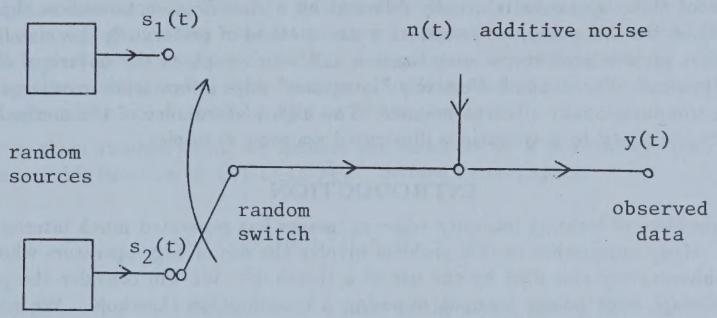


Figure: Scholtz's Model for Sawchuck's Problem

SEQUENTIAL EDGE LINKING[1]

PAUL H. EICHEL* and EDWARD J. DELP**
*Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109
**School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

The use of edge operators to enhance intensity edges in images is commonplace. The application of these operators is usually followed by a classification procedure that relies on a threshold. In this paper, we introduce a new method of performing this classification that does not involve such coarse quantization and which exploits the nature of edges to aid in the process. The method effectively "integrates" edge information over large sets of pixels in a computationally efficient manner. The high performance of the method in the case of very low signal-to-noise ratio is illustrated via some examples.

INTRODUCTION

The problem of locating intensity edges in images has generated much interest in the literature. Many approaches to this problem involve the use of edge operators whose outputs are subsequently classified by the use of a threshold. We will consider the problem of linking image edge points without imposing a classification threshold. We will view this as a post - processing task to follow the application of an edge operator. For the present, we consider an edge operator to be any mathematical operator such that, when applied to a digital image, produces an estimate of the magnitude and direction of the intensity gradient at every point in the image. The gradient magnitude and direction information represents the input to our system. Our goal is to determine, with as much positional accuracy as possible, connected lines representing intensity edges in the original image while not responding to spurious effects of noise, etc.

A crucial observation is that it is not desirable to quantize this gradient information any more coarsely than necessary. We certainly do not wish to threshold this data (quantize to one bit) globally, locally, adaptively, or with hysteresis. Nor do we wish to non - maximum suppress it [1]. The experience of detection theory, coding theory, and related disciplines indicates that coarse quantization of noisy, correlated observations independently of one another may be expedient or acceptable when the signal - to - noise ratio is high, but leads to poor performance when the signal quality deteriorates (cf. [2],[3]). Instead of making such hard decisions (thresholding) right away, experience dictates one "integrate" the soft information over many observations before concluding on a decision. Sequential Edge Linking (SEL) is one manner in which such integration may be performed in the context of image edge detection.

Integrating edge information over a large set of pixels becomes increasingly difficult to do with long, directional edge operator masks as the number of pixels in the mask increases. This is because, for large operator masks, the number of possible edge configurations becomes enormous. Even with certain restrictions imposed, the number of possible configurations grows exponentially with the number of pixels in the operator.

[1]This work was partially supported by the Center for Robotics and Integrated Manufacturing, Robot Systems Division, College of Engineering, University of Michigan, and the Air Force Office of Scientific Research under Grant F49620-82-C-0089.

It is tempting to suggest a maximum likelihood approach such as [4]. To do this, one might hypothesize all possible edge paths of, say, n pixels in length, calculate their respective a-priori likelihoods, assuming some distribution on the gradient levels, and pick the largest. However, the exponential growth of the candidate configurations with n confounds this approach as well, regardless of the maximum likelihood algorithm used. With Sequential Edge Linking, one does not attempt to explore all possible paths in the tree, picking the ML path, but rather one explores only a small subset of them with a high probability of including the correct one. This exploration is accomplished sequentially, extending only those paths which hold high promise of being the correct one.

DEFINITIONS

1.) Images, Random Fields.

For the purposes of this paper, the term image will refer to a two - dimensional discrete parameter random field [5]. Sample functions of this random field consist of a rectangular array of numbers. Whatever their interpretation, these numbers, henceforth to be called pixel values or pixels, are defined at points of a rectangular array. The points of the array are called nodes, their spacing is uniform and equal in both directions, and they are indexed by I × I where I is the set of integers.

For a given random field, we assume the existence of a probability space (Ω, A, P) and a measurable function, $f_{\vec{r}}(\omega)$; $\omega \in \Omega$, $\vec{r} \in I^2$ defined on this space:

$$f_{\vec{r}}(\omega) : \Omega \times I^2 \rightarrow R \tag{1}$$

Here, $\vec{r} = (r^1, r^2)$ is the coordinates of a node in the array, and $f_{\vec{r}}(\omega)$ is the pixel value at that node.

Due to the rectangular nature of the array, each node has a unique set of eight neighbors [6]. Any ordered set of nodes for which all adjacent pairs are neighbors will be called connected. An ordered pair of neighbors also uniquely defines a direction. Eight such directions are thus possible.

2.) Paths.

A path will be defined as a connected set of nodes with the following property: for any subset of three nodes in the ordered set, the directions defined by the first two nodes and by the second two nodes differ by $\pi / 4$ or less. Thus, if one were to draw line segments between nodes of a path, the resulting curve would contain no angles of $\pi / 2$ or more acute.

Paths may be denoted in one of two ways. The first is simply the ordered set of nodes comprising the path:

$$\mathbf{m} = [(r_1^1, r_1^2), (r_2^1, r_2^2), \ldots (r_n^1, r_n^2)] \tag{2}$$

The second method is by specifying a start node and direction, and an ordered set of letters:

$$\mathbf{m} = (r_0^1, r_0^2) \times \vec{d}_0 \times [a_1, a_2, a_3, \ldots a_n] \tag{3}$$

where (r_0^1, r_0^2) is the start node, $\vec{d}_0$ is the start direction, and the a_i are taken from an alphabet A = [L,S,R]. The letters of the alphabet stand for left, straight, and right respectively. The coordinates of the nodes of the path may be obtained in a recursive

fashion from the letters a_i. The first node is obtained from the start node by moving in the array in the direction $\vec{d}_0$. The direction to the second node is then obtained from $\vec{d}_0$ and a_1 and so on by the obvious rule:

$$\vec{d}_1 = \begin{cases} \vec{d}_0 + \frac{\pi}{4} & \text{if } a_1 = L \\ \vec{d}_0 & \text{if } a_1 = S \\ \vec{d}_0 - \frac{\pi}{4} & \text{if } a_1 = R \end{cases} \tag{4}$$

This process can likewise be reversed to obtain the list - of - letters description of a path from the list of node coordinates. Thus, either description uniquely specifies a given path and they are equivalent.

This second path description has been introduced in order to clarify the following model for paths. We will assume that paths may be modeled as a k^{th} order Markov Chain [7]. Let us consider a discrete time stochastic process S_i, i = 0,1,2,3, . . . :

$$S_i\ (\omega) : \Omega \rightarrow \Sigma \tag{5}$$

where the underlying space Ω is just the space of all possible state sequences: $\Omega = (s_0, s_1, s_2, \ldots)$ and Σ is the state space. When the process enters the state at i+1 from that at i, it outputs a letter $a_{i+1} \in A = [\,L,S,R\,]$. The state of the system is defined to be the ordered sequence of the last k letters output by the system:

$$s_i = (a_i, a_{i-1}, \ldots, a_{i-(k-1)}) \tag{6}$$

So that $\Sigma = A \times A \times \ldots \times A = A^k$. We will assume s_0 is fixed and known. The Markov assumption is then:

$$Pr\,(\,S_{i+1} = s_{i+1} \mid s_i, s_{i-1}, \ldots, s_0\,) = Pr\,(\,S_{i+1} = s_{i+1} \mid s_i\,) \tag{7}$$

from which it follows by the chain rule:

$$\begin{aligned} Pr(\mathbf{m}) &= Pr(a_1, a_2, \ldots, a_n) \\ &= Pr(s_n \mid s_{n-1})\, Pr(s_{n-1} \mid s_{n-2}) \ldots Pr(s_1 \mid s_0) \end{aligned} \tag{8}$$

This is our stochastic model for the path process. It bears a strong resemblance to the model used in [4]. By manipulation of the transition probabilities, $Pr(s_{i+1} \mid s_i)$, certain types of paths achieve higher probability of occurrence than do others.

3.) Path Metric.

The crucial concept in any sequential tree searching algorithm is that of a path metric. The algorithm's efficiency and chance of success lie in its ability at any point of execution to rank all the paths explored so far by a measure of their "quality". This measure is the path metric. It was first introduced in the problem of sequential decoding of convolutional codes by R. M. Fano [8].

Our path metric will be defined for paths in a random field as defined in preceding sections. We will assume the following two measures on the random field. The first, $p_1(f_{\vec{r}} = y)$, is the probability the value of the field at node $\vec{r}$ is y conditioned on the hypothesis, H_1, that that node is on an intensity edge in the original image:

$$\begin{aligned} p_1(f_{\vec{r}} = y) &= Pr(f_{\vec{r}} = y \mid H_1) \\ &= Pr(f_{\vec{r}} = y \mid \vec{r} \textit{ is on an edge }) \end{aligned} \tag{9}$$

The second is the probability of the value of the field conditioned on the null hypothesis, H_0, that the node is randomly positioned in the field with respect to intensity edges:

$$\begin{aligned} p_0(f_{\vec{r}} = y) &= Pr(f_{\vec{r}} = y \mid H_0) \\ &= Pr(f_{\vec{r}} = y \mid \vec{r} \textit{ is a random node }) \end{aligned} \tag{10}$$

Note that, at this point, we are not even specifying what the random field, f, represents but merely assuming the existence of these two conditional measures on that field.

The approach used to derive our path metric is similar to that used by Massey in the convolutional coding situation [9]. The principal idea is to employ "random tails" appended to the ends of a set of hypothesized paths to make them all the same length. The best path in the set is then chosen on the basis of a rule that optimizes a joint probability. This procedure insures that the resulting decision statistic or "metric" is not biased by path length but rather chooses the best path solely on the basis of its probability of occurance.

Let us suppose we are given a set, M, of possible paths of varying lengths,

$$\begin{aligned} M &= [\mathbf{m}^1, \mathbf{m}^2, \ldots, \mathbf{m}^k] \\ \mathbf{m}^i &= \vec{r}_0^{\,i} \times \vec{d}_0^{\,i} \times [a_1^i, \ldots, a_{n^i}^i] \end{aligned} \tag{11}$$

where the length of the i^{th} path, $\mathbf{m}^i$, is n^i. As before, the i^{th} path has probability:

$$Pr(\mathbf{m}^i) = \prod_{j=1}^{n^i} Pr(s_j^i \mid s_{j-1}^i) \tag{12}$$

Of course, each node of every path in M has an associated pixel value. To simplify notation, we will denote the value of the j^{th} node of the i^{th} path as: $f_{\vec{r}_j^{\,i}} = f_j^i$. The set of f-values associated with the i^{th} path will be denoted: $\mathbf{f}^i = [f_1^i, f_2^i, \ldots, f_{n^i}^i]$. We will assume these f-values are statistically independent given H_1. Just as in [9], we will append a string of a-letters and associated f-values to the end of the shorter paths to make them all the same length. The tail of a-letters, $\mathbf{m}_{tail}^i = [a_{n^i+1}^i, \ldots, a_N^i]$ where N is the length of the longest path in M, are chosen at random. Since this path extension is random through the field, the associated f-values are chosen independently and at random according to the measure, $p_0(f)$. We assume the values of the random tails are chosen independently of everything else, so that:

$$Pr(f^i_{n'+1}, \ldots, f^i_N \mid \mathbf{m}^i, \mathbf{m}^i_{tail}, \mathbf{f}^i) = Pr(f^i_{n'+1}, \ldots, f^i_N) = \prod_{j=n'+1}^{N} p_0(f^i_j) \tag{13}$$

If we now assume the probabilities of the f-values of any path that coincides with an intensity edge are given by the measure, p_1, then due to the conditional independence of the observations, we have:

$$Pr(\mathbf{f}^i, \mathbf{f}^i_{tail} \mid H_1) = \prod_{j=1}^{n'} p_1(f^i_j) \quad \prod_{j=1}^{N-n'} p_0(f^i_{n'+j}) \tag{14}$$

Thus the joint probability of tracing path $\mathbf{m}^i$, observing the values $\mathbf{f}^i$ along it, and adding a random tail, $\mathbf{m}^i_{tail}$, under the hypothesis that path i is indeed an edge path may be written:

$$Pr(\mathbf{m}^i, \mathbf{f}^i, \mathbf{m}^i_{tail}) = \prod_{j=1}^{n'} Pr(s^i_j \mid s^i_{j-1}) \prod_{j=1}^{n'} p_1(f^i_j) \prod_{j=1}^{N-n'} Pr(s^i_{n'+j} \mid s^i_{n'+j-1}) \prod_{j=1}^{N-n'} p_0(f^i_{n'+j}) \tag{15}$$

Summing over all possible choices of a random tail, we have:

$$\begin{aligned} Pr(\mathbf{m}^i, \mathbf{f}^i) &= \sum_{\mathbf{m}^i_{tail}} Pr(\mathbf{m}^i, \mathbf{f}^i, \mathbf{m}^i_{tail}) \\ &= \prod_{j=1}^{n'} Pr(s^i_j \mid s^i_{j-1}) \prod_{j=1}^{n'} p_1(f^i_j) \prod_{j=1}^{N-n'} p_0(f^i_{n'+j}) \end{aligned} \tag{16}$$

Now an optimum rule for deciding among the paths in M is to choose that path i such that i maximizes $Pr(\mathbf{m}^i, \mathbf{f}^i)$ or equivalently, such that i maximizes:

$$\ln \frac{Pr(\mathbf{m}^i, \mathbf{f}^i)}{\prod_{j=1}^{N} p_0(f_j)} \tag{17}$$

since the denominator is independent of the path $\mathbf{m}^i$ and the logarithm is a monotonic function. We thus have as the statistic to be maximized:

$$\Gamma(\mathbf{m}^i, \mathbf{f}^i) = \sum_{j=1}^{n'} \left[\ln \frac{p_1(f^i_j)}{p_0(f^i_j)} + \ln Pr(s^i_j \mid s^i_{j-1}) \right] \tag{18}$$

We see that the metric given by Equation (18) has no built in length bias: that path which maximizes the joint probability exhibits the highest metric. Just as in [9], the use of the random tail is a convenient device for equalizing the length of the observed paths for the purposes of comparing their probabilities.

The conditional mean of the statistic, $\frac{\Gamma(\mathbf{m},\mathbf{f})}{n}$, i.e. the path metric normalized by its length provides some insight into the usefulness of the metric for edge linking. Under the

H_1 hypothesis, this expected value is:

$$E\left\{\frac{\Gamma(\mathbf{m},\mathbf{f})}{n} \mid H_1\right\} = E\left\{\frac{1}{n}\sum_{j=1}^{n}\left[\ln\frac{p_1(f_j)}{p_0(f_j)} + \ln Pr(s_j \mid s_{j-1})\right] \mid H_1\right\} \tag{19}$$

Now since the path process is stochastically independent of the random field under the H_1 hypothesis, this becomes:

$$E\left\{\frac{\Gamma(\mathbf{m},\mathbf{f})}{n} \mid H_1\right\} = E\left\{\frac{1}{n}\sum_{j=1}^{n}\left[\ln\frac{p_1(f_j)}{p_0(f_j)}\right] \mid H_1\right\} + E\left\{\frac{1}{n}\sum_{j=1}^{n}\left[\ln Pr(s_j \mid s_{j-1})\right]\right\} \tag{20}$$

Considering the first term above, since the observations, f_j, are independent, the expectation of the sum is equal to the sum of n identical expectations, so that:

$$\begin{aligned} E\left\{\frac{1}{n}\sum_{j=1}^{n}\left[\ln\frac{p_1(f_j)}{p_0(f_j)}\right] \mid H_1\right\} &= E\left\{\ln\frac{p_1(f)}{p_0(f)} \mid H_1\right\} \\ &= \sum_f p_1(f)\ln\frac{p_1(f)}{p_0(f)} \end{aligned} \tag{21}$$

by the definition of the p_1 measure. This last quantity is seen to be the Kullback's Information between the p_1 and p_0 measures, denoted $I(p_1 \mid p_0)$ [10].

The second term of Equation (20) can be rewritten using the chain rule as:

$$\begin{aligned} E\left\{\frac{1}{n}\sum_{j=1}^{n}\left[\ln Pr(s_j \mid s_{j-1})\right]\right\} &= -E\left\{\frac{1}{n}\ln\frac{1}{Pr(s_n, s_{n-1}, \ldots, s_1)}\right\} \\ &= -E\left\{\frac{1}{n}\ln\frac{1}{Pr(a_n, a_{n-1}, \ldots, a_1)}\right\} \end{aligned} \tag{22}$$

which is defined to be the entropy per letter of the Markov source [11]. In the limit as $n \to \infty$, this quantity can be shown [11] to be equal to:

$$-H_\infty(S) = -\sum_{i=1}^{3^t} q(i)\sum_{l=1}^{3} Pr(a_l \mid s = i)\ln\frac{1}{Pr(a_l \mid s = i)} \tag{23}$$

where $q(i)$ is the time averaged state probability of state i:

$$q(i) = \lim_{n\to\infty}\frac{1}{n}\sum_{j=1}^{n} Pr(s_j = i) \tag{24}$$

Since all the states in the Markov Chain we have been considering are recurrent and in the same equivalence class, i.e. each state can be reached from any other in one or more transitions and no state outside this set can be reached, $q(i)$ is independent of the initial state, s_0 [7]. Combining Equations (21)-(23), we therefore have the result that:

$$\lim_{n\to\infty} E\left\{\frac{\Gamma(\mathbf{m},\mathbf{f})}{n} \mid H_1\right\} = I(p_1 \mid p_0) - H_\infty(S) \tag{25}$$

Likewise, under the H_0 hypothesis, the observations term becomes:

$$\begin{aligned} E\left\{\frac{1}{n}\sum_{j=1}^{n}\left[\ln\frac{p_1(f_j)}{p_0(f_j)}\right] \mid H_0\right\} &= E\left\{\ln\frac{p_1(f)}{p_0(f)} \mid H_0\right\} \\ &= \sum_f p_0(f)\ln\frac{p_1(f)}{p_0(f)} \\ &= -I(p_0 \mid p_1) \end{aligned} \tag{26}$$

so that:

$$\lim_{n\to\infty} E\left\{\frac{\Gamma(\mathbf{m},\mathbf{f})}{n} \mid H_0\right\} = -I(p_1 \mid p_0) - H_\infty(S) \tag{27}$$

We see then that since both $I(\cdot \mid \cdot)$ and $H_\infty(M)$ are strictly non-negative, the expected value of the normalized metric conditioned on the hypothesis that the path is not an edge path is negative. When conditioned on the H_1 hypothesis, the normalized metric will be positive if:

$$I(p_1 \mid p_0) > H_\infty(S) \tag{28}$$

Therefore, as long as Equation (28) is satisfied, the metric tends to grow with length if the path is truly an edge path and tends to decrease with length otherwise. This is the desired "drift" tendency that allows the searching algorithm to find the edges in the image. It is interesting to note that the difference between the statistics conditioned on the two hypotheses grows with n as:

$$\begin{aligned} E\left\{\frac{\Gamma(\mathbf{m},\mathbf{f})}{n} \mid H_1\right\} - E\left\{\frac{\Gamma(\mathbf{m},\mathbf{f})}{n} \mid H_0\right\} &= I(p_1 \mid p_0) - I(p_0 \mid p_1) \\ &= J(p_0,p_1) \end{aligned} \tag{29}$$

where $J(\cdot,\cdot)$ is Kullback's Divergence [10]. This says that, on average, the difference between the rate at which the metric grows along a true edge path and the rate at which it decreases along a random path is equal to the Kullback's Divergence between the two measures p_1 and p_0.

SEQUENTIAL SEARCHING

We come now to the important question of sequential searching. So far, we have defined what we mean by paths and can associate with each path a quantitative measure of quality or metric. This metric is a function of the joint probability of the path letters and the f-values of the random field along the path.

What is now needed is a systematic way of hypothesizing paths so that they may be compared via their corresponding metrics. As was brought out earlier, every path in the field cannot be hypothesized because the number of possibilities is enormous. Thus, maximum likelihood detection is impossible. What we do instead is make hypotheses in such a way that only a small subset of possible paths is actually explored, but this subset contains an actual edge path with high probability.

The procedure used is called sequential tree searching and is borrowed from coding theory [12]. It presumes that a start or root node is given. This start node must be on an intensity edge but is otherwise arbitrary. The selection of start nodes will not be discussed here. The root node defines a tree of semi-infinite paths in the random field. Because of the special structure of paths, there are precisely 3^n paths of length n (or to depth n in the tree) beginning at the root node. The sequential searching algorithm begins at the root node and sequentially examines paths in the tree. The exact procedure by which this is done depends on the algorithm used. The sequential behavior, however, is common to them all. At each iteration of the algorithm, one of the paths explored so far is extended by one node and the metric for this new path is calculated. Decisions are made based on these metrics as to which path should be extended on the next iteration.

Because long paths are built up node by node, it is entirely possible that the algorithm can be tricked by noisy f-values into incorrectly following a non-edge path for some depth into the tree. Earlier, we saw that the average tendency of the metric under these circumstances is to decrease with increasing length. One hopes therefore that these forays along incorrect paths will eventually be halted by the resulting decreases in metric. The algorithm at some point settles back onto the correct (edge) path for which the average tendency is for the metric to increase with length. Of course, this average behavior of the metric does not guarantee that all incorrect paths will eventually be abandoned. Such questions can be formulated and answered in a quantitative manner, but this analysis shall be addressed in a subsequent paper.

There are a number of candidate algorithms for performing this tree searching. They include the Fano algorithm [8], the Zigangirov - Jelinek (Z-J) algorithm or stack algorithm [12], the A^* algorithm [13], and various modifications of the above such as the stack-bucket and multiple-stack algorithms [12]. The examples given in the next Section were all processed using the Z-J algorithm, but any of the others might have been used instead. The interested reader is referred to the cited references for thorough discussions on their operation and characteristics.

EXAMPLES

The Sequential Edge Linking technique was applied to several images by way of example, and the resulting edges are displayed in the Figures. In the case of the first two images, the input to the sequential searching algorithm was two random fields: the gradient magnitude and the gradient direction as estimated by a pair of 3x3 operator masks, one oriented along the x-axis and the other along the y-axis. Both the magnitude and direction fields were quantized to 8 bits per pixel. The gradient direction information was incorporated into the path metrics by forming the inner product of the normal to the gradient direction at a node with the path direction at that node and multiplying the gradient magnitude by the result. Thus the f-field values entering the metric were actually the directional gradients normal to the path direction. Figure 1 is an image of connecting rods taken under normal lighting conditions. Only start nodes on the outside contour were given; interior edges could have been found but were ignored in this example. Figure 2 is an arterial angiogram and exhibits some very poor signal-to-noise ratio and low-contrast edges. Note in both examples the one pixel wide and completely connected nature of the detected edges that result from SEL.

The input to the SEL algorithm for the third image was also the gradient magnitude and direction, but in this case the image was first smoothed by a large Gaussian filter ($\sigma = 3$ pixels). This is the so-called Pratt Figure of Merit test image [14] at a signal-to-noise ratio of -7 dB, where SNR is defined to be:

$$SNR = 10 \log_{10} \frac{h^2}{\sigma^2}$$

and h is the height of the intensity step. The Figure of Merit for this 128x128 image was 1.0.

The last image was processed in similar fashion to the third. It is a Pratt Figure of Merit - like image in which the intensity edge is square, forcing the algorithm to negotiate corners. Due to the Markov edge model (which had an identical transition matrix for all four examples), this is slightly more difficult than the single straight edge of Figure 3. The SNR for Figure 4 is -3dB and the merit value is 0.82. The starting nodes for Figures 3 and 4 were simply the nodes with highest gradient magnitude.

REFERENCES

1 J. Canny, "Finding edges and lines in images," *MIT AI-TR-720,* June 1983.

2 M. Schwartz, *Information, Transmission, Modulation and Noise,* McGraw-Hill, 1980, p. 460.

3 J. Melsa and D. Cohn, *Decision and Estimation Theory,* McGraw-Hill, 1978, p. 134.

4 D. Cooper, "Maximum likelihood estimation of markov-process blob boundaries in noisy images, *IEEE Trans. PAMI,* vol. PAMI-1, pp. 372-384, Oct. 1979.

5 E. Wong, *Stochastic Processes in Information and Dynamical Systems,* McGraw-Hill, 1971, p. 243.

6 A. Rosenfeld, "Connectivity in digital pictures," *J. ACM,* vol. 17, pp. 146-160, Jan. 1970.

7 S. Ross, *Stochastic Processes,* Wiley, 1983.

8 R. Fano, " A hueristic discussion of probabilistic decoding," *IEEE Trans. Inf. Theory,* vol. IT-9, pp. 64-74, Apr. 1963.

9 J. Massey, "Variable - length codes and the Fano metric," *IEEE Trans. Inf. Theory,* vol. IT-18, pp. 196-198, Jan. 1972.

10 S. Kullback, *Information Theory and Statistics,* Wiley, 1959.

11 R. Gallager, *Information Theory and Reliable Communication,* Wiley, 1968.

12 S. Lin and D. Costello Jr., *Error Control Coding: Fundamentals and Applications,* Prentice-Hall, Englewood Cliffs N.J., 1983.

13 N. Nilsson, *Problem-Solving Methods in Artificial Intelligence,* McGraw-Hill, 1971, pp. 54-70.

14 W. Pratt, *Digital Image Processing,* Wiley-Interscience, 1978, p. 495.

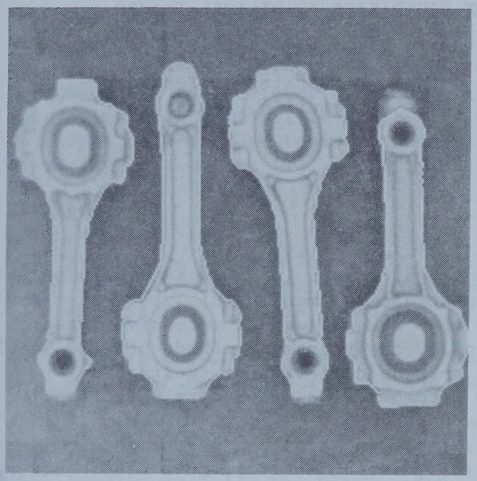


Figure 1. An eight bit gray level image of connecting rods. The exterior boundaries found by the Sequential Edge Linking Algorithm are indicated by the white overlay.

Figure 2. An eight bit gray level image of an arterial angiogram, overlaid with boundaries detected by SEL.

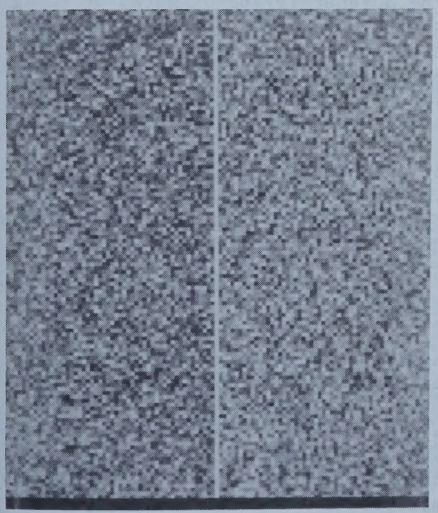


Figure 3. A Pratt Figure of Merit test image with a vertical edge and SNR = -7 dB. SEL detected edge is indicated with overlay. Merit value is 1.0.

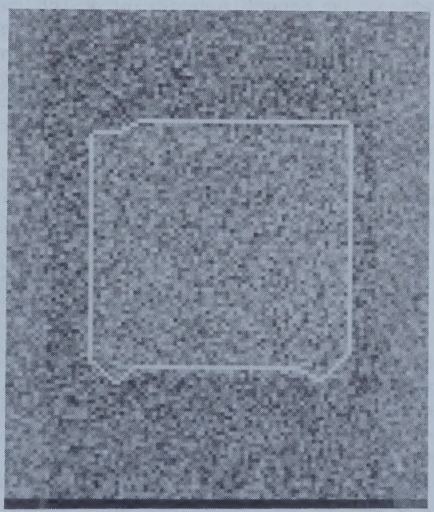


Figure 4. A Pratt Figure of Merit test image with a box edge and SNR = -3 dB. Merit value is 0.82.

SPLINE BASED SEQUENTIAL ESTIMATION OF BOUNDARIES IN DIGITAL IMAGES

ANTONIOS LIAKOPOULOS
System Dynamics Inc.
Gainesville, Florida

DRAGANA BRZAKOVIC
University of Tennessee
Knoxville, Tennessee

Boundaries delimiting objects and their parts in an image are recognized to be a crucial link between an image and its meaningful interpretation. Consequently, boundary detection has been the subject of extensive research during the last decade. Most frequently, the proposed methods utilize local measurements of image properties and rely on detection of their abrupt changes. Only recently, it has been recognized that boundary detection is essentially an estimation problem[1,2].

This paper describes the design, implementation and performance of a recursive boundary estimator for digital monochromatic images. The boundary is globally estimated using a linear recursive filter algorithm and a set of local measurements. Boundary estimation is preceeded by image labeling[1] which results into the identification of image regions exhibiting homogeneous properties and the intermediate zones where boundaries lie (Figure 1). It is assumed that each boundary can be described by a single-valued function f(y). Smooth boundaries with no jumps or corner discontinuities are obtained by a recursive approximation scheme which uses cubic spline functions while nonsmooth boundaries are obtained by utilizing piecewise linear approximation or cubic splines under tension[3]. The emphasis in this work is on smooth boundaries which can be adequately approximated by cubic splines on an interval [0,a] with knots at $0=y_1 < y_2 \ldots < y_{n+1} = a$. The twice differentiable cubic spline function g(y) approximating f(y) in the interval $0 \le y \le a$ is explicitely determined in terms of $f_j (j=1,2,\ldots,n+1)$, f'_1 and f'_{n+1}, where f_i and f'_i denote function and derivative values at the i-th knot. Consequently, the state vector of the system is an (n+3)-dimensional vector, $\underline{x}$.

The uncertainty in the system model is reduced by considering measurements performed in subregion Q, the intermediate zone between the homogeneous regions R_1 and R_2 (Figure 1). The measurement process is modeled as

$$z = \frac{p_1 \delta + (\rho - \delta) p_2}{\rho} + \varepsilon \qquad (1)$$

where p_1 and p_2 denote the known values that a suitable property, p, takes in regions R_1 and R_2 respectively, ε measurement error to be stochastically modeled and the lengths ρ and δ are defined in Figure 1. The choice of property p depends strongly on the type of images being processed. The average gray level is the simplest available property but it is of limited applicability. For complex images containing textured surfaces utilization of G-matrix approach[1] has been found appropriate. To enhance the system observability, m measurements are concurrently processed.

A sensitivity analysis of the boundary estimation accuracy as affected by variations in different algorithm and system parameters has been conducted. Preliminary results have shown that the model is relatively insensitive to underestimated or overestimated values of the process and noise covariance matrix elements although unrealistic values of these parameters may cause filter divergence.

The proposed method has been evaluated on images with controlled noise

level as well as images containing texture regions with unknown noise characteristics. For both image categories, the results obtained are in very good agreement with human perception. An example of the obtained results is shown in Figure 2. The filter converges rapidly and the procedure is particularly well suited for real time applications. The study shows that the first order autoregresive process is adequate in modeling the error ε appearing in equation (1).

REFERENCES

(1) Brzakovic, D., and Tou, J.T., "Boundary Determination of Object Surfaces Via Textural Information," Seventh International Conference on Pattern Recognition, Montreal, Canada, July 30-August 2, 1984.

(2) Chen, P.C., and Pavlidis, T., "Image Segmentation as an Estimation Problem," Computer Graphics and Image Processing, Vol. 12, No. 2, pp. 153-172, 1980.

(3) Pruess, S., "Properties of Splines in Tension", Journal of Approximation Theory, Vol. 17, pp. 86-96, 1976.

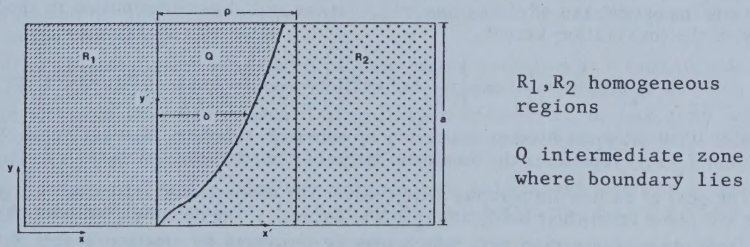


Figure 1. Boundary determination nomenclature.

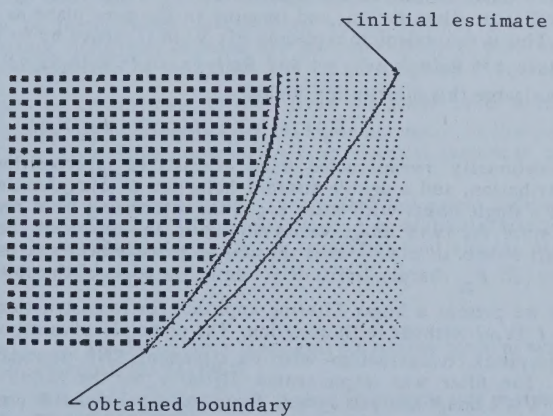


Figure 2. An example of the obtained results.

RECONSTRUCTION OF REAL 2-D SIGNALS FROM THEIR IN-LINE HOLOGRAMS BY APPROXIMATE INVERSE FILTERING

LEVENT ONURAL and PETER D. SCOTT
Electrical and Computer Engineering Department
State University of New York at Buffalo
Amherst, N.Y. 14260

Abstract

A 2D linear filtering approach to the in-line hologram reconstruction problem is presented and its effectiveness demonstrated on real and synthesized holograms. Implemented as a digital filter operating on sampled data, optical reconstruction is bypassed and twin image interference is removed.

Summary

Constituent equations describing in-line holography have been previously derived using the principles of physical optics [1,2]. Predicted images based on this model agree closely with direct densitometer measurements [2]. In a form convenient for the present purpose, the central result of this model may be expressed as

$$I_z(x,y) = 1 - 2\,a(x,y) \;**\; \mathrm{Re}\{h_z(x,y)\} \tag{1}$$

where z is the optical axis of the system, $a(x,y)$ is the real object distribution in the $z = 0$ plane, $h_z(x,y)$ is the convolution kernel,

$$h_z(x,y) = \frac{1}{j\lambda z}\exp\left[j\frac{\pi}{\lambda z}(x^2+y^2)\right] \overset{\mathcal{F}}{\longleftrightarrow} \exp\left[-j\frac{\lambda z}{4\pi}(u^2+v^2)\right] \tag{2}$$

associated with Huygens-Fresnel scattering of coherent radiation at wavelength λ onto the z plane, and $I_z(x,y)$ represents the dominant terms of the resulting hologram.

The goal of in-line holography is to recover the crossection distribution $a(x,y)$ of objects in the z=0 plane from their hologram $I_z(x,y)$. Typically, the hologram includes object distributions from other planes than z=0, which may be recovered by creating images $R_z(x,y)$ which reconstruct the object volume one crossection at a time. Here we consider objects restricted to the z=0 plane.

Conventional reconstruction is performed by reilluminating the developed hologram with the same coherent illumination, and imaging in the same plane as the hologram was originally recorded. This is equivalent to replacing $a(x,y)$ in (1) above by its hologram $I_z(x,y)$ multiplied by -1. Since $1 ** \mathrm{Re}\{h_z(x,y)\} = 1$ and $\mathrm{Re}\{h_z(x,y)\} ** \mathrm{Re}\{h_z(x,y)\} = \tfrac{1}{2}\delta(x,y) + \tfrac{1}{2}\mathrm{Re}\{h_{2z}(x,y)\}$, after some algebra this substitution yields:

$$R_z(x,y) = 2 - 2a(x,y) + I_{2z}(x,y). \tag{3}$$

This conventionally reconstructed image consists of superposition of a dc term, the desired object distribution, and a double-distance hologram of $a(x,y)$ called the twin image. If $a(x,y)$ consists of a single object with sharp edges, the interfering twin image does not hamper particle detection since the twin image varies slowly in the region of the particle itself. If there are many small objects in $a(x,y)$, as is the case in particle field analysis (a principle use of in-line holography [2]), I_{2z} sharply limits detection of the edges of the objects comprising $a(x,y)$.

Here we present a linear filtering approach for reconstruction of $a(x,y)$ directly from its hologram $I_z(x,y)$ without reillumination. The principle advantages of this approach are that it bypasses physical reconstruction with its attendant SNR degradation, and the twin image is removed. The filter was implemented digitally on the SUNY Buffalo LAPES Laboratory COMTAL/VAX image analysis sytem. Sampling aspects of this problem are discussed elsewhere [3]. If the impulse response $\mathrm{Re}\{h_z(x,y)\} = (1/\lambda z)\sin(\pi/\lambda z)(x^2+y^2)$ in (1) possesed an inverse, $a(x,y)$ could be recovered from (1) by linear inverse filtering. The system function (2D Fourier transform) is:

$$H(u,v) = \cos\frac{\lambda z}{4\pi}(u^2+v^2) \tag{4}$$

which, having uncountably infinite zeros, is clearly the the system function of a noninvertable

filter. Formally expanding $(\cos q)^{-1}$ in a Fourier series and replacing q by $(\lambda z / 4\pi)(u^2+v^2)$ we define a series expansion

$$H^{-1}(u,v) = 2 \sum_{k=0}^{\infty} (-1)^{-k} \cos\left[(2k+1)\frac{\pi}{\lambda z}(u^2+v^2)\right] \tag{5}$$

By discrete convolution with a suitable convergence sequence or "window function", this divergent series is converted into a convergent one in R^2 [4]. The resulting series, finitely truncated, is then a smoothed approximation to the inverse filter:

$$H^{-1}(u,v) = \cos\frac{\pi}{\lambda z}(u^2+v^2) + \sum_{n=1}^{2^{M-1}-1} \left\{\frac{M-1-\lfloor \log_2 n \rfloor}{M}\right\} S_n(u,v) \tag{6}$$

where

$$S_n(u,v) = (-1)^n \cos\frac{\pi}{\lambda z}(2n+1)(u^2+v^2). \tag{7}$$

Application of this filter to the hologram yields a dc term, an object reconstruction term proportional to $a(x,y)$, and a residual error term. Even for small values of M this error is substantially less than the error arising from conventional reconstruction (twin image).

Comments

1. The first term in $H^{-1}(u,v)$ corresponds to conventional reconstruction (if "real part" operation is used instead of "magnitude square"), thus the remainder may be interpreted as successive reductions in the twin image.

2. The effect of the subsequent terms is to produce partial outputs which each include $a(x,y)$ superposed on $I_{kz}(x,y)$, i.e. a twin image interference term with expanded spatial axes. In the superposition of terms over n, the desired $a(x,y)$ appear invariantly in each term while the interfering terms are not in-phase. Essentially, this system averages outputs in which the desired signal $a(x,y)$ is coherent and the undesired terms are not.

3. The interpretation in 2. above leads to a very compact and efficient algorithmic realization, which was implemented on the COMTAL/VAX system to generate images shown in the talk.

4. While $\mathrm{Re}\{h_z(x,y)\}$ is not invertible, $h_z(x,y)$ is. An alternative derivation of $H^{-1}(u,v)$ proceeds as follows: if $I_z(x,y)$ is convolved with $h_z^{-1}(x,y)$, the result may be divided into real and imaginary parts. The real part contains $a(x,y)$ plus twin image, while the imaginary part is similar to the double distance hologram $I_{2z}(x,y)$. The imaginary part may now be treated as another hologram and can be reconstructed as explained above, yielding a part containing $a(x,y)$ plus the double distance hologram, and another part containing a quadruple distance hologram-like term. This procedure can be continued, and the parts having $a(x,y)$ are averaged.

5. $H^{-1}(u,v)$ is an edge-preserving approximate inverse. Even the lowest order terms distribute energy over all frequency ranges. Thus truncation of H^{-1} does not result in low-pass smearing of the edges of the objects in $a(x,y)$. Edge preservation is an essential feature in particle field analysis.

6. Computational experience with experimental and synthesized holograms indicates that the main benefit of this filter is realized after 3-10 terms. Beyond a relatively small number of terms the twin image is barely visible, and cumulating truncation and aliasing errors reduces the benefit of further iterations.

References

1 G.B. Parrent, B.J. Thompson, "On the Fraunhofer (far field) diffraction patterns with coherent background", Optica Acta 11, 183-93 (1964).

2 G.A. Tyler, B.J. Thompson, "Fraunhofer holography applied to particle size analysis -a reassessment" Optica Acta 23, 685-700 (1976).

3 G. Liu, P.D. Scott, D.T. Shaw, "Modelling and sampling strategies for digital in-line holography", Proc. OSA Annual Meeting (1984) (to appear)

4 P.L. Butzer, R.J. Nessel, "Fourier analysis and approximation", vol.1, Academic Press, (1971).

A HOLOGRAM-LIKE DIGITAL REPRESENTATION OF PICTURES

PRZEMYSLAW PRUSINKIEWICZ
University of Regina
Regina, Saskatchewan, Canada

ABSTRACT

A new representation of pictures suitable for their representation with progressive resolution, is outlined.

SUMMARY

This short paper describes a new representation of pictures which is suitable for their transmission with progressive resolution [1,2,4]. The main feature of this representation is: Given a data stream Y, representing a picture with full resolution, various (not necessarily initial) segments of Y represent the same picture with appropriately lower resolutions. This is analogous to the well-known feature of holograms: Any portion of a hologram represents the same picture that the whole hologram does, but with a lower resolution. Other digital representations of pictures described in the literature do not have this feature.

The new representation is based on a picture traversal algorithm (i.e. the selection and ordering of sampling points) with the following key property: Given a string of sampling points $P=\langle p_0,p_1,p_2,\ldots\rangle$, various substrings of P consist of points uniformly distributed in sampling region Q. Hence, the longer a substring of corresponding sampled values is, the better is the resolution of the reconstruction of the original picture.

The essential notions are formalized below.

Definition 1. Let T>0 denote the edge size of the square sampling region Q(T) (its vertices are: (0,0), (0,T), (T,T), (T,0).) Moreover, let the binary word $r_{2n+1}r_{2n}\cdots r_1r_0$ represent index k of a sample point $P_k=(x_k,y_k)$ in the pure binary number system:

$$k = \sum_{i=0}^{2n+1} r_i 2^i \quad .$$ The coordinates of point P_k are then defined as:

$$x_k = T \sum_{i=0}^{n} r_{2i} 2^{-(i+1)}$$

$$y_k = T \sum_{i=0}^{n} r_{2i+1} 2^{-(i+1)}$$

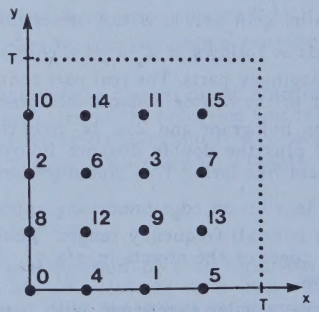


Fig. 1.

Example 1. The first sixteen points of string P are shown in Fig. 1.

Definition 2. Given a sampling region Q(T), a sampling lattice of mxm elements, with the origin in point (c_x,c_y), is the set:

$$M_{mxm}(c_x,c_y) = \{(c_x+i\tfrac{T}{m},\ c_y+j\tfrac{T}{m}):\ i,j=0,1,\ldots,m-1\}$$

The sampling lattice formalizes the notion of mxm points uniformly distributed in the square Q(T).

Example 2. The set $\{p_0,\ldots p_{15}\}$ (Fig. 1), as well as its subsets $\{p_0,p_1,p_2,p_3\}$, $\{p_4,p_5,p_6,p_7\}$, etc., form sampling lattices in square Q(T). Theorem 1 shows, that this is not accidental.

Theorem 1. Let $\hat{P}(n,h)$ denote the following subset of 4^n elements of string P:

$$\hat{P}(n,h) = \{p_{h\cdot 4^n}, p_{h\cdot 4^n+1}, \ldots, p_{(h+1)4^n-1}\}$$

For any n,h = 0,1,2,..., the set $\hat{P}(n,h)$ is a sampling lattice in region Q(T):

$$(\forall n,h = 0,1,2,\ldots)(\exists c_x,c_y \in [0,\frac{T}{2^n}))\ \hat{P}(n,h) = M_{2^n x 2^n}(c_x,c_y)$$

For an outline of the proof see [3].

Conclusion. Consider a string Y of samples of a picture f in points $p_0,p_1,p_2,\ldots$ The substring Y(n,h) of samples of f corresponding to the set of sampling points $\hat{P}(n,h)$ makes it possible to reconstruct f with the resolution increasing with n (the length of the substring) and independent of h (the position of the substring in the string).

Example 3 (due to Mark Christopher). Examples of the reconstruction of a picture (consisting of a circle, a straight line, and a rectangle) from substrings of samples are shown in Fig. 2.

There are also other strings of sampling points satisfying Theorem 1. Formal properties of the entire class of the hologram-like representations, and their practical applications (videotex, browsing through pictorial data bases, etc.) are being studied.

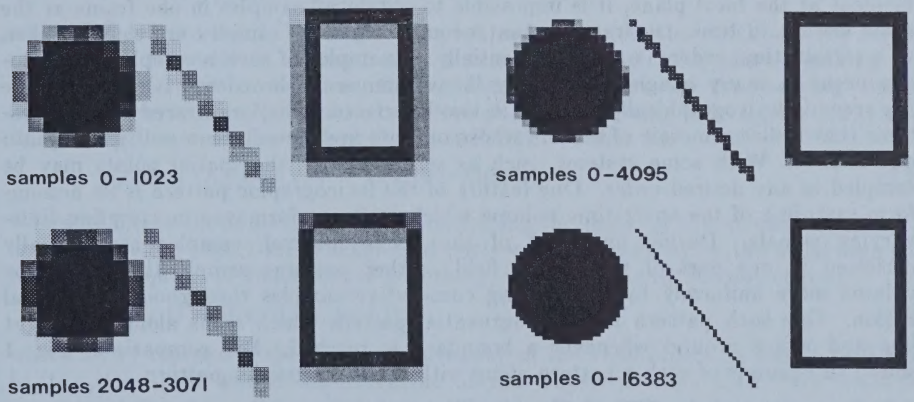


Fig. 2.

ACKNOWLEDGEMENT

This research was supported by a grant from Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] K. Knowlton: Progressive transmission of gray-scale and B/W pictures by simple, efficient and lossless encoding schemes. IEEE Proceedings 68 (1980) pp. 885-896.

[2] T. Pavlidis: Algorithms for graphics and image processing. Computer Science Press 1982.

[3] P. Prusinkiewicz: A hologram-like digital representation of pictures, Tech. Rep. CS-84-01, Dept. of Comp. Sci., University of Regina 1984.

[4] K.R. Sloan, S.L. Tanimoto: Progressive refinement of raster images. IEEE Trans. Comput. C-28 (1979) pp. 871-874.

RECONSTRUCTION OF 2-D SIGNALS FROM IRREGULARLY SPACED SAMPLES WITH AN APPLICATION TO THE RECOVERY OF TIME-SEQUENTIALLY SAMPLED MOVING OBJECTS *

DAVID S. CHEN†, *JAN P. ALLEBACH*
School of Electrical Engineering, Purdue University, West Lafayette, IN 47907

ABSTRACT

A model for reconstruction of moving objects from samples obtained time-sequentially is studied. A minimum mean-squared error estimator of the object is established via projection theory. The relation between the mean-squared error and the set of irregularly spaced sampling points is studied using the minimax principle. Three criteria for evaluating the performance of the sampling point set are derived. These criteria are used to evaluate scan patterns for time-sequentially sampling a moving object.

1.0 INTRODUCTION

1.1 Time-Sequential Sampling

Digital processing of time-varying imagery has numerous industrial, medical and military applications which include target tracking and identification, pattern recognition and robot vision. In the absence of a separate channel for each spatial sampling point or some means of instantaneously storing a record of the radiation incident at the focal plane, it is impossible to obtain all samples in one frame at the same instant in time, i.e. frame instantaneously. Instead, samples are usually taken in a certain time order, i.e. time-sequentially. Examples of such a sampling mechanism occur in many imaging systems such as commercial broadcast television where the scene is lexicographically scanned in two interlaced fields, or infrared imaging systems that utilize a mosaic of sensors whose outputs are time-division multiplexed into one channel. With some systems, such as sensor arrays, the spatial points may be sampled in any desired order. One feature of the lexicographic pattern is its nonuniform sampling of the space-time volume which limits performance in sampling time-varying signals. During any part of the frame interval, samples are spatially clustered in one part of the spatial field. Other patterns sample the space-time volume more uniformly by distributing consecutive samples throughout the spatial region. One such pattern is the congruential pattern which scans along a straight line and wraps around whenever a boundary is reached. For comparison, Fig. 1 shows an example of such a pattern along with the lexicographic pattern.

1.2 Reconstruction of Time-Varying Images

In this paper, we will study the reconstruction of time-varying images from samples obtained time-sequentially. In particular, we consider the case of a rigid object undergoing translational motion along a known trajectory. During the frame interval, the object moves a substantial distance relative to the size of the spatial field. When projected back to a non-time-sequential 2-D lattice, the samples form a blurred version of the moving object. Since the trajectory is known, a deblurred version of the object can be recovered by displacement correction. After the displacement correction, the samples are in their correct spatial locations; however, they are no longer equally spaced. The procedure is illustrated in Fig. 2. In this case, the Whittaker-Kotelnikov-Shannon sampling theorem [1] is not applicable because of the irregularity in spacing and the fact that only a finite number of samples are available.

*This work was supported by the National Science Foundation under Grant ESC-8312231.
†Now with the Computer Science Department, General Motors Research Laboratories, Warren, MI 48090-0955.

Reconstruction of 1-D signals from nonuniformly spaced samples has been studied by Yen [2], Levi [3], Yao [4], Papoulis [5], Beutler [6], and Higgins [7]. In our approach, we assume an estimator of the form $\hat{f}(\vec{x}) = \sum_{l=0}^{L-1} b_l \psi(\vec{x}-\vec{x}_l)$, where $\psi(\vec{x})$ is a bandlimited interpolating function and the $\vec{x}_l$ are the irregularly spaced sampling points. Applying the Hilbert space projection theory, we solve for the b_l that make this a minimum mean-squared error (MMSE) estimate of $f(\vec{x})$. The 1-D version of the estimator coincides with one obtained earlier [2] - [4]. The minimum mean-squared error also depends on the set of irregularly spaced sampling points $(\vec{x}_0,\vec{x}_1, \ldots ,\vec{x}_{L-1})$. We attempt to further minimize this error by proper choice of the point set. To avoid signal dependence, we consider a minimax approach. Three criteria are derived for the evaluation of sampling point sets. These criteria are applied to evaluate scan patterns for time-sequentially sampling a moving object. The major features that distinguish our work from that of previous authors are the emphasis on minimizing the mean-squared error in reconstructing a bandlimited signal from a finite number of irregularly spaced samples, the analysis of the dependence of this error on the sampling point set, and the application of these results to the recovery of time-sequentially sampled moving objects.

2.0 BACKGROUND

2.1 Signal Representation

Throughout this paper, we consider a 2-D rectangular field of view R:

$$R = \{(x,y)\in \mathbb{R}^2 \mid 0 \le x \le A_1, 0 \le y \le A_2 \}, \tag{2.1}$$

where $\mathbb{R}^2$ denotes the 2-D real space, and $A_1$and A_2 denote the width and the height of the field, respectively. Using vector notation $\vec{x} = (x,y)'$ with prime denoting the transpose of vectors or matrices, the signal $g(\vec{x},t)$ indicates the time-varying image intensity observed on R. It consists of two parts, the moving object part $O(\vec{x},t)$ and the possibly time-varying background part $B(\vec{x},t)$. We thus have that

$$g(\vec{x},t) = O(\vec{x},t)M(\vec{x},t) + B(\vec{x},t)(1-M(\vec{x},t)), \tag{2.2}$$

where $M(\vec{x},t)$ is the indicator function indicating the mask of the moving object. We assume that the object and background parts can be separated by the usual segmentation techniques. Without loss of generality, we consider only the object part

$$O(x,y,t) = f(x-d_x(t), y-d_y(t)), \tag{2.3}$$

where $f(x,y)$ represents the unknown intensity of the rigid object at time $t = 0$ and $(d_x(t),d_y(t))$ is the known displacement of this object at time $t>0$. We want to reconstruct the object $f(\vec{x})$ from samples taken on R. Initially, the object may lie outside the field of view; so $f(\vec{x})$ is defined on an extended region which we take to be $\mathbb{R}^2$.

2.2 Time-Sequential Sampling

We consider a rectangular lattice in our sampling process. Let X and Y be the sampling intervals along the horizontal and vertical directions. Let M and N be two integers such that $A_1 = MX$, $A_2 = NY$. Thus, all the sampling points on R are given by a set S_R of pairs:

$$S_R = \{ (mX, nY),\ m = 0,1,\ldots,M-1,\ n = 0,1,\ldots,N-1 \}. \tag{2.4}$$

The total number of sampling points on R is $L=MN$. These L points are sampled time-sequentially according to an order determined by the sampling pattern

$$S = \{ (\alpha_l, \beta_l) \mid (\alpha_l, \beta_l)\in S_R,\ l = 0,1,\ldots,L-1;\ \text{and } (\alpha_l,\beta_l) \ne (\alpha_k,\beta_k),\ l \ne k. \}, \tag{2.5}$$

where the pair (α_l, β_l) is the spatial location of the l-th sampling point in the

sequence. If we let T_R be the set containing all time-sequential sampling patterns, it is easy to see that T_R has L! elements. In some cases, the sampling pattern can be expressed as a closed form function of l. For example, with the lexicographic pattern,

$$\begin{aligned} \alpha_l &= \lfloor l/N \rfloor X, \\ \beta_l &= (l \bmod N)Y, \end{aligned} \tag{2.6}$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

The congruential sampling patterns C_R form a subclass of T_R consisting of patterns of the form [2]:

$$C(\alpha,\beta) = \{ (\alpha_l, \beta_l) \mid \alpha_l = (\alpha l \bmod M)X, \beta_l = (\beta l \bmod N)Y;\ l = 0,1,...,L-1 \}, \tag{2.7}$$

where the pair (α,β) must satisfy

$$\gcd(M,N)=1,\ \gcd(\alpha,M)=1 \text{ and } \gcd(\beta,N)=1. \tag{2.8}$$

Here gcd(x,y) denotes the greatest common divisor of x, y. In Fig. 1, M=5 and N=7; and the congruential pattern is $C(2,3)$.

2.3 Displacement Correction

For a given sampling pattern, the time-sequentially sampled version of the moving object can be written as:

$$h(x,y,t) = \sum_{l=0}^{L-1} f(x-d_x(t),y-d_y(t))\delta(x-\alpha_l, y-\beta_l, t-lT), \tag{2.9}$$

where T is the time interval between successive samples, and $\delta(x,y,t)$ is a 3-D Dirac function. In order to process the signal in 2-D space, we project the samples of h(x,y,t) to the 2-D spatial plane at t=0. To deblur the object, we then employ a displacement correction procedure under the assumption that the displacement $(d_x(t),d_y(t))$ is known. Let h(x,y) be the deblurred version of the samples obtained:

$$h(x,y) = \sum_{l=0}^{L-1} f(x,y)\delta(x-x_l, y-y_l), \tag{2.10}$$

where $(x_l, y_l) = (\alpha_l-d_x(lT),\beta_l-d_y(lT))$ denotes the irregular sampling points yielded by the displacement correction. Using vector notation, we can write (2.10) as

$$h(\vec{x}) = \sum_{l=0}^{L-1} f(\vec{x}_l)\delta(\vec{x}-\vec{x}_l). \tag{2.11}$$

3.0 THE MINIMUM MEAN SQUARED ERROR ESTIMATOR

3.1 A Generalized Sampling Theorem

We now seek to recover $f(\vec{x})$ from the samples $f(\vec{x}_0),f(\vec{x}_1),...,f(\vec{x}_{L-1})$. Although perfect restoration may be impossible, it is nevertheless desirable to reproduce $f(\vec{x})$ as closely as possible. Let us consider an estimator of $f(\vec{x})$ that is of the general form

$$\hat{f}_L(\vec{x}) = \sum_{l=0}^{L-1} b_l\psi(\vec{x}-\vec{x}_l), \tag{3.1}$$

where b_l is a function of the samples $f(\vec{x}_0),f(\vec{x}_1),...,f(\vec{x}_{L-1})$, and $\psi(\vec{x})$ is the unknown interpolating function. It is natural to think of $f(\vec{x}) - \hat{f}_L(\vec{x})$ as an error which is to be minimized in some sense. Using Hilbert spaces, the minimum mean-squared error criterion associated with projection theory is the obvious choice.

Let **F** be the space consisting of 2-D square integrable Ω–bandlimited functions $f(\vec{x})$:

$$\mathbf{F} = \{ f(\vec{x}) \mid \int_{R^2} |f(\vec{x})|^2 d\vec{x} < \infty; \quad \text{and} \quad F(\vec{u}) = 0, \quad \vec{u} \notin \Omega \}, \tag{3.2}$$

where $F(\vec{u})$ is the Fourier transform of f, $\vec{u}=(u,v)'$ and $\int_{\mathbf{R}^2} d\vec{x} = \int\int_{\mathbf{R}^2} dxdy$ is the two-fold integral. The relation between f and F is given by the following two integrals:

$$f(\vec{x}) = \int_{\Omega} F(\vec{u})\exp[i2\pi\vec{x}\cdot\vec{u}]\, d\vec{u} \quad . \tag{3.3}$$

$$F(\vec{u}) = \int_{\mathbf{R}^2} f(\vec{x})\exp[-i2\pi\vec{x}\cdot\vec{u}]\, d\vec{x} \quad . \tag{3.4}$$

F is a Hilbert space with inner product defined as:

$$(f_1, f_2) = A_{\Omega}\int_{\mathbf{R}^2} f_1(\vec{x})\overline{f_2}(\vec{x})\, d\vec{x}, \qquad f_1, f_2 \in \mathbf{F}, \tag{3.5}$$

where A_{Ω} is the area of the band region Ω and $\overline{f}$ denotes the complex conjugate of f. Let the norm $\|f\|$ be defined as $\sqrt{(f,f)}$. We aim to establish an estimator $\hat{f}_L(\vec{x})$ that minimizes the mean-squared error

$$e_f^2 = \|f - \hat{f}_L\|^2. \tag{3.6}$$

Assume that $\psi \in \mathbf{F}$ and let $\mathbf{V}(e_L)$ represent the linear span of $e_L = \{\ \psi(\vec{x}-\vec{x}_l),\ l = 0,1,...,L-1\ \}$. Then $\hat{f}_L \in \mathbf{V}(e_L) \subset \mathbf{F}$. From the projection theorem, we obtain the following set of normal equations for $\hat{f}_L$:

$$(\ f-\hat{f}_L,\ \psi(\vec{x}-\vec{x}_m)\ ) = 0,\ \ m = 0,1,...,L-1. \tag{3.7}$$

By the linearity of the inner product, we expand (3.7):

$$(\ f,\ \psi(\vec{x}-\vec{x}_m)\ ) = \sum_{l=0}^{L-1} b_l(\ \psi(\vec{x}-\vec{x}_l),\ \psi(\vec{x}-\vec{x}_m)\ ),\quad m = 0,1,...,L-1. \tag{3.8}$$

By the definition of inner product and using the Parseval equality, we can write the inner products in (3.8) as

$$A_{\Omega}\int_{\mathbf{R}^2} f(\vec{x})\ \overline{\psi(\vec{x}-\vec{x}_m)}\, d\vec{x} = A_{\Omega}\int_{\Omega} F(\vec{u})\ \overline{\Psi(\vec{u})}\ \exp[i2\pi\vec{x}_m\cdot\vec{u}]\, d\vec{u}, \tag{3.9}$$

$$A_{\Omega}\int_{\mathbf{R}^2} \psi(\vec{x}-\vec{x}_l)\ \overline{\psi(\vec{x}-\vec{x}_m)}\, d\vec{x} = A_{\Omega}\int_{\Omega} |\ \Psi(\vec{u})|^2\ \exp[-i2\pi(\vec{x}_l-\vec{x}_m)\cdot\vec{u}]\, d\vec{u}, \tag{3.10}$$

where $\Psi(\vec{u})$ is the Fourier transform of $\psi(\vec{x})$. The fact that $f(\vec{x})$ is known only for L sample values $f(\vec{x}_0), f(\vec{x}_1), ..., f(\vec{x}_{L-1})$ makes it impossible to compute the integral (3.9) for a general function ψ. We then choose ψ to be a particular function ϕ such that its Fourier transform Φ is the indicator function of the band region Ω normalized by its area A_{Ω}:

$$\Phi(\vec{u}) = \begin{cases} 1/A_{\Omega}, & \vec{u}\in\Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

By the Fourier inverse relationship, (3.9) is then equal to the sample value $f(\vec{x}_m)$ and (3.10) is equal to $\phi(\vec{x}_m-\vec{x}_l)$. Note that with this normalization, $\phi(0)=1$. Substituting these results into (3.8), we obtain:

$$f(\vec{x}_m) = \sum_{l=0}^{L-1} b_l\phi(\vec{x}_m-\vec{x}_l),\quad m = 0,1,...,L-1, \tag{3.12}$$

which shows that $\hat{f}_L$ has the same value as f at the sampling points $\vec{x}_l$. This is the appealing interpolating property we wish to have.

Using vector notation, (3.12) can be written as

$$\vec{\mathbf{f}} = \mathbf{\Phi}\vec{\mathbf{b}}\ , \tag{3.13}$$

where $\vec{\mathbf{f}} = [f(\vec{x}_0), f(\vec{x}_1), \ldots, f(\vec{x}_{L-1})]'$, $\vec{\mathbf{b}} = [b_0, b_1, \ldots, b_{L-1}]'$ and Φ is an L×L matrix with (l,m)-th element $\phi(\vec{x}_m - \vec{x}_l)$. The Fourier transform of $\phi(\vec{x} - \vec{x}_l)$ is $\Phi(\vec{u})\exp[-i2\pi\vec{x}_l \cdot \vec{u}]$. The elements in $E_L = \{ \Phi(\vec{u})\exp[-i2\pi\vec{x}_l \cdot \vec{u}],\ l = 0,1,\ldots,L-1 \}$ are linearly independent. From the linearity of the Fourier transform, it follows that the elements in $e_L = \{ \phi(\vec{x} - \vec{x}_l),\ l = 0,1,\ldots,L-1 \}$ are also linearly independent. Therefore, the matrix Φ is nonsingular. We thus have a unique solution for $\vec{\mathbf{b}}$:

$$\vec{\mathbf{b}} = \Phi^{-1}\vec{\mathbf{f}}. \tag{3.14}$$

The matrix Φ is also Hermitian by the fact that $\phi_{lm} = \overline{\phi_{ml}}$. For a symmetric band region Ω, i.e., $\Omega(u,v) = \Omega(-u,-v)$, Φ is symmetric. We can further show that Φ is positive definite.

Proposition 3.1

The L × L matrix Φ defined above is positive definite.

Proof: Consider $\vec{u}$ as a random vector uniformly distributed over the finite region Ω. Then the vector $\vec{\eta} = (\exp[-i2\pi\vec{x}_l \cdot \vec{u}],\ l = 0,1,\ldots,L-1)'$ is a random vector on $\mathbb{R}^L$ with each element being a function of $\vec{u}$. Let the covariance matrix of $\vec{\eta}$ be $\Sigma_{\vec{\eta}} = E\{\vec{\eta}\vec{\eta}'\}$. The (l,m)-th element of the covariance matrix is

$$E\{\exp[-i2\pi\vec{x}_l \cdot \vec{u}]\exp[i2\pi\vec{x}_m \cdot \vec{u}]\} = \frac{1}{A_\Omega}\int_\Omega \exp[-i2\pi(\vec{x}_l - \vec{x}_m)\cdot\vec{u}]d\vec{u},$$

$$= \phi(\vec{x}_m - \vec{x}_l).$$

Hence $\Phi = \Sigma_{\vec{\eta}}$. Since the covariance matrix is semipositive definite and Φ is nonsingular, the conclusion of the proposition holds. (q.e.d.)

So far we have obtained a closed form for $\hat{f}_L$ which is the projection of f on $\mathbf{V}(e_L)$. The above arguments are summarized in the following theorem.

Theorem 3.2

Let $f(\vec{x})$ be an Ω-bandlimited square integrable function on $\mathbb{R}^2$. Let $(\vec{x}_l)$ be L irregularly spaced distinct sampling points. Let $\mathbf{V}(e_L) = \mathbf{V}\{\phi(\vec{x} - \vec{x}_l),\ l = 0,1,\ldots,L-1\}$, where $\phi(\vec{x})$ is the inverse Fourier transform of the band region Ω. Then there exists a MMSE estimator $\hat{f}_L$ of f in $\mathbf{V}\{e_L\}$ such that

$$\|f - \hat{f}_L\|^2 = \inf\|f - g\|^2,\quad g \in \mathbf{V}(e_L). \tag{3.15}$$

Further,

$$\hat{f}_L(\vec{x}) = \sum_{l=0}^{L-1} b_l \phi(\vec{x} - \vec{x}_l), \tag{3.16}$$

with $\vec{\mathbf{b}} = [b_0, b_1, \ldots, b_{L-1}]'$ given by

$$\vec{\mathbf{b}} = \Phi^{-1}\vec{\mathbf{f}}. \tag{3.17}$$

Here, $\vec{\mathbf{f}} = [f(\vec{x}_0), f(\vec{x}_1), \ldots, f(\vec{x}_{L-1})]'$; and Φ is an L×L matrix with (l,m)-th element $\phi(\vec{x}_m - \vec{x}_l)$.

We now consider two important properties of the estimator obtained in Theorem 3.2.

Proposition 3.3 Interpolating Property

The estimator $\hat{f}_L(\vec{x}_l)$ has the same values as $f(\vec{x}_l)$ at the sampling points $(\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_{L-1})$; so $\hat{f}_L(\vec{x}_l) = f(\vec{x}_l)$, $l = 0,1,\ldots,L-1$.

Proof: See (3.12).

Proposition 3.4 Minimum Energy Property

Let $\mathbf{G}_L$ be the class of Ω-bandlimited functions that have value $f(\vec{x}_l)$ at $\vec{x}_l$, $l = 0,1,...,L-1$, then $\|\hat{f}_L\|^2 = \inf_{g \in G_L} \|g\|^2$.

Proof: For any g in $\mathbf{G}_L$, it follows from Theorem 3.2 that $\hat{f}_L$ is its MMSE estimator. In terms of projection theory, the difference $e_g = g - \hat{f}_L$ is orthogonal to $\hat{f}_L$. Thus, $\|g\|^2 = \|e_g\|^2 + \|\hat{f}_L\|^2$, from which it follows that $\|g\|^2 \geq \|\hat{f}_L\|^2$.

4.0 DEPENDENCE OF ESTIMATOR PERFORMANCE ON THE IRREGULARLY SPACED SAMPLING POINTS

4.1 The Minimax Principle

We have developed a mathematical model to study the optimal reconstruction of a 2-D signal f from a finite set $\chi = \{\vec{x}_0, \vec{x}_1, ..., \vec{x}_{L-1}\}$ of irregularly spaced sampling points. The optimality is in the sense of minimizing the mean-squared error $\mathbf{e}_f^2(\chi)$ for a given set of sampling points. The mean-squared error will depend on both the signal and the locations of the irregularly spaced sampling points. Intuitively, we expect the error to decrease as the sampling points become more uniformly spaced. In this section, we consider how the mean-squared error varies with the set χ of sampling points, independently of the signal. We do this by considering the maximum of the mean-squared error over a class of signals. We say that a point set χ_1 is preferred to a point set χ_2 if

$$\max_f \left[\mathbf{e}_f^2(\chi_1)\right] \leq \max_f \left[\mathbf{e}_f^2(\chi_2)\right] \tag{4.1}$$

We then choose the set χ from an allowable class $\mathbf{X}$ of such point sets that minimizes this maximum error. We thus seek the minimax point set χ_{op} which minimizes $\max_f \mathbf{e}_f^2(\chi)$ among all sets in $\mathbf{X}$, i.e.,

$$\max_f \mathbf{e}_f^2(\chi_{op}) = \min_{\chi} \max_f \mathbf{e}_f^2(\chi). \tag{4.2}$$

Before we prove an important theorem in this section, we first give a simpler version of the mean-squared error.

Proposition 4.1

Using the notation in Sec. 3, we have

$$\mathbf{e}_f^2 = \|f - \hat{f}_L\|^2 = \|f\|^2 - \|\hat{f}_L\|^2, \tag{4.3}$$

$$= \|f\|^2 - \vec{\mathbf{f}}' \mathbf{\Phi}^{-1} \vec{\mathbf{f}}.$$

Proof: From the proof of the minimum energy property, we only need to show that $\|\hat{f}_L\|^2 = \vec{\mathbf{f}}' \mathbf{\Phi}^{-1} \vec{\mathbf{f}}$. By the linearity of the inner product and (3.10), we have

$$\|\hat{f}_L\|^2 = \sum_{l=0}^{L-1} \sum_{m=0}^{L-1} b_l b_m \langle \phi(\vec{x} - \vec{x}_l), \phi(\vec{x} - \vec{x}_m) \rangle,$$

$$= \sum_{l=0}^{L-1} \sum_{m=0}^{L-1} b_l b_m \phi(\vec{x}_l - \vec{x}_m) = \vec{\mathbf{b}}' \mathbf{\Phi} \vec{\mathbf{b}}.$$

From (3.14), $\vec{\mathbf{b}} = \mathbf{\Phi}^{-1} \mathbf{f}$; hence $\|\hat{f}_L\|^2 = \vec{\mathbf{f}}' \mathbf{\Phi}^{-1} \mathbf{\Phi} \mathbf{\Phi}^{-1} \vec{\mathbf{f}} = \vec{\mathbf{f}}' \mathbf{\Phi}^{-1} \vec{\mathbf{f}}$. (q.e.d.)

Because the matrix $\mathbf{\Phi}$ is uniquely determined by the point set χ, there is a one to one correspondence between χ and $\mathbf{\Phi}$. We thus call the matrix $\mathbf{\Phi}$ the point set matrix. Equations (4.1) and (4.2) can then be restated as follows. A point set χ_1 is preferred to a point set χ_2 if $\max_f \mathbf{e}_f^2(\mathbf{\Phi}_1) \leq \max_f \mathbf{e}_f^2(\mathbf{\Phi}_2)$, where $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ are the point set matrices corresponding to χ_1 and χ_2. Similarly, a point set χ_{op} is minimax

if its point set matrix Φ_{op} satisfies $\max_{f} e_f^2(\Phi_{op}) = \min_{\Phi} \max_{f} e_f^2(\Phi)$. Since we want to separate the dependence of e_f^2 on the point set χ from its dependence on f, it is natural, in view of (4.3), to search for the minimax solution among a class of signals for which $\|f\|$ is constant and a class of point sets for which $\|\vec{f}\|$ is constant. The following theorem gives a criterion for the minimax point set with respect to these two classes in terms of point set matrices.

Theorem 4.2

Over a class of signals with constant energy $\|f\|^2$ and a class of point sets for which $\|\vec{f}\|^2$ is constant, the minimax point set χ_{op} satisfies

$$\lambda_{max}(\Phi_{op}) = \min_{\Phi} \lambda_{max}(\Phi), \tag{4.4}$$

where $\lambda_{max}(\Phi)$ is the maximum eigenvalue of Φ.

Proof: Using Proposition 4.1 and letting $\|f\|^2 = M$, $\max_{f} e_f^2(\Phi) = \max_{f} \{\|f\|^2 - \vec{f}^* \Phi^{-1} \vec{f}\} = M - \min_{f} \vec{f}^* \Phi^{-1} \vec{f}$. Minimizing $\max_{f} e_f^2(\Phi)$ is then equivalent to maximizing $\min_{f} \vec{f}^* \Phi^{-1} \vec{f}$ over Φ. Without loss of generality, we can assume $\|\vec{f}\|^2 = 1$. Because Φ is positive definite, its inverse is also; thus $\min_{f} \vec{f}^* \Phi^{-1} \vec{f} = \lambda_{min}(\Phi^{-1})$. Since the eigenvalues of Φ^{-1} are reciprocals to the eigenvalues of Φ, i.e., $\lambda_{min}(\Phi^{-1}) = [\lambda_{max}(\Phi)]^{-1}$, the minimax criterion becomes $\min_{\Phi} \left[\lambda_{max}(\Phi)\right]$. (q.e.d.)

Theorem 4.2 leads to the criterion that point set χ_1 is preferred to χ_2 if $\lambda_{max}(\Phi_1) \leq \lambda_{max}(\Phi_2)$.

4.2 Two Additional Criteria of Point Set Goodness

The order of the point set matrix Φ equals the total number of sampling points. For good resolution, this number is usually quite large. This makes it difficult, if not impossible, to calculate its eigenvalues. Upon this consideration, we develop some criteria related to $\lambda_{max}(\Phi)$ that are simpler to evaluate. We first prove a lemma and then establish a bound for $\lambda_{max}(\Phi)$

Lemma For the point set matrix Φ, we have

$$\lambda_{min} \leq 1 \leq \lambda_{max}. \tag{4.5}$$

Proof: This is easily seen from the inequality $\lambda_{min} \leq \frac{1}{L} \sum_{l=0}^{L-1} \lambda_l \leq \lambda_{max}$, and the fact that trace $\Phi = \sum_{l=0}^{L-1} \lambda_l = L$. (q.e.d.)

Applying the lemma to the bound [9], $\max_{i,j} |\lambda_i - \lambda_j| \leq \|\Phi\|$, we obtain

$$\lambda_{max}(\Phi) \leq \sqrt{2}\,\|\Phi\| + 1 . \tag{4.6}$$

Thus, to minimize $\lambda_{max}(\Phi)$, we minimize $\|\Phi\|$. This leads to our second criterion that χ_1 is preferred to χ_2, if $\|\Phi_1\|^2 \leq \|\Phi_2\|^2$.

Next, we apply some results in matrix theory on the localization of eigenvalues to give another criterion for the goodness of χ. By the Levy-Desplanques Theorem [9], $\det(\Phi) \neq 0$ if $\phi_{ii} > Q_i$, where $Q_i = \sum_{j \neq i} |\phi_{ij}|$. Using this theorem, we can show

Proposition 4.4

All eigenvalues of the matrix Φ lie in the closed interval

$$|\lambda - 1| \leq Q(\Phi), \tag{4.7}$$

where $Q(\Phi) = \max_{i} \left[Q_i\right]$, and $Q_i = \sum_{j \neq i} |\phi_{ij}|$.

Proof: Let λ_r be an eigenvalue of Φ. Then $\det(\lambda_r \mathbf{I} - \Phi) = 0$; and by the Levy-Desplanques Theorem and the fact that $\phi(0) = 1$, $|\lambda_r - 1| \leq Q_i$ for at least one i. Thus $|\lambda_r - 1| \leq Q(\Phi)$.

This proposition states that all the eigenvalues lie in an interval centered at 1 with length equal to $2Q(\Phi)$. To minimize the maximum eigenvalue over Φ, we minimize $Q(\Phi)$ over Φ. Hence we have the third criterion that a point set χ_1 is preferred to χ_2 if $Q(\Phi_1) \leq Q(\Phi_2)$.

5.0 EVALUATION OF PATTERNS FOR TIME-SEQUENTIALLY SAMPLING A MOVING OBJECT

To compare the three criteria for evaluation of point sets, and gain insight on the characteristics of good and bad point sets as ranked by these criteria, we consider the problem of recovering an object moving with constant velocity along a straight line across a field of view containing a 5 × 7 array of sampling points. When a lexicographic ordering is used, the point set shown in Fig. 3 will result after displacement correction. In general, a different point set will be obtained for each sampling pattern used. All these point sets will be contained within the region of visibility which depends on the trajectory. In the interest of uniformly resolving this region, we choose a rectangular bandregion Ω corresponding to uniform distribution of the 35 samples over it. In addition to the lexicographic pattern, we evaluated 24 different congruential patterns. Table 1 shows the values of the three criteria for these patterns ranked according to $\lambda_{max}(\Phi)$. Figure 4 shows the 25 point sets also ranked in this order. We see that $\lambda_{max}(\Phi)$ ranks the patterns in order of decreasing uniformity. Although there is only loose agreement between the rankings under the three criteria, $\lambda_{max}(\Phi)$ for the patterns ranked highest by $||\Phi||$ and $Q(\Phi)$ is within 5% of its minimum value.

6.0 CONCLUSION

We have studied a method of interpolating a finite number of irregularly spaced samples of a bandlimited signal. We derived the MMSE estimator of the form $\hat{f}_L(\vec{x}) = \sum_{l=0}^{L-1} b_l \phi(\vec{x} - \vec{x}_l)$. The interpolating function $\phi(\vec{x})$ is the normalized inverse Fourier transform of the band region; and the coefficients b_l satisfy the set of linear equations $\Phi\vec{b} = \vec{f}$. Here Φ is the point set matrix determined by the set of sampling points; and $\vec{f}$ is the vector consisting of sample values. The energy of the estimator $\hat{f}_L(\vec{x})$ is the minimum over all Ω-bandlimited signals that interpolate the sample values $f(\vec{x}_l)$ at $\vec{x}_l$. Since the mean-squared error of the estimator is a function of the point set, we then considered the performance of the point set $(\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_{L-1})$ in signal reconstruction. To eliminate dependence on the signals, we considered a minimax approach. We found that the optimum point set is the one for which the maximum eigenvalue of the point set matrix is smallest. Two additional criteria for the evaluation of point sets where derived via matrix theory. They are the norm $||\Phi||$ of the point set matrix, and the maximum sum $Q(\Phi)$ of the absolute values of the off-diagonal row elements. These criteria are valuable because they are much more easily computed than the maximum eigenvalue. Finally, these criteria were applied to the particular problem of evaluating sampling patterns for reconstructing an object undergoing translation along a linear trajectory. In this case, the criteria favor a set of sampling points distributed as nearly as possible on an uniform rectangular lattice.

REFERENCES

[1] A. J. Jerri, "The Shannon Sampling Theorem -- Its Various Extensions and Applications: A Tutorial Review," *Proc. IEEE,* Vol. 65, pp. 1565-1596, Nov. 1977.

[2] J. L. Yen, "On Nonuniform Sampling of Bandlimited Signals," *IRE Trans. on Circuit Theory,* Vol. CT-3, pp. 251-257, Dec. 1956.

[3] L. Levi, "Fitting a Bandlimited Signal to Given Points," *IEEE Trans. Inform. Theory,* Vol. IT-11, pp. 372-376, July 1966.

[4] K. Yao, "Applications of Reproducing Kernel Hilbert Spaces -- Bandlimited Signal Models," *Inform. and Contr.,* Vol. 11, pp. 429-444, 1967.

[5] A. Papoulis, "Error Analysis in Sampling Theory," *Proc. IEEE,* Vol. 54, pp. 947-955, July 1966.

[6] F. J. Beutler, "Error-free Recovery of Signals from Irregularly Spaced Samples," *SIAM Review,* Vol. 8, pp. 328-335, July 1966.

[7] J. R. Higgins, "A Sampling Theorem for Irregularly Spaced Sample Points," *IEEE Trans. Inform. Theory,* Vol. IT-21, pp. 621-622, Sept. 1976.

[8] J. P. Allebach, "Design of Antialiasing Patterns for Time-Sequential Sampling of Spatiotemporal Signals," *IEEE Trans. Acoust., Speech, Signal Processing,* Vol. ASSP-32, pp. 137-144, Feb. 1984.

[9] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities,* Boston: Allyn and Bacon, 1964, pp. 146-167.

LEXICOGRAPHIC

β	α 0	1	2	3	4
6	30	31	32	33	34
5	25	26	27	28	29
4	20	21	22	23	24
3	15	16	17	18	19
2	10	11	12	13	14
1	5	6	7	8	9
0	0	1	2	3	4

CONGRUENTIAL

β	α 0	1	2	3	4
6	30	23	16	9	2
5	25	18	11	4	32
4	20	13	6	34	27
3	15	8	1	29	22
2	10	3	31	24	17
1	5	33	26	19	12
0	0	28	21	14	7

Fig. 1. Time-sequential sampling patterns for a 5×7 field of view. The entries give the order in which the points (α,β) are sampled.

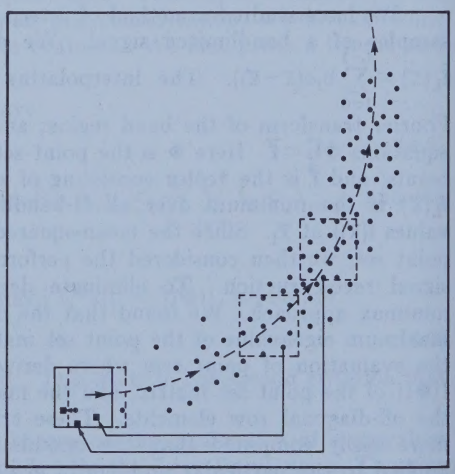


Fig. 2. One frame of a moving square object sampled time-sequentially with a CGR-DI pattern. Displacement correction is illustrated for three samples.

TABLE 1. CRITERIA FOR THE SAMPLING PATTERNS

Rank	Pattern	$\lambda_{max}(\Phi)$	$\|\|\Phi\|\|$	$Q(\Phi)$
1	$C(3,5)$	2.2745	7.3925	2.4471
2	$C(1,6)$	2.2783	7.5388	3.1509
3	$C(1,5)$	2.3032	7.4627	2.3884
4	$C(3,6)$	2.3177	7.1697	2.8977
5	$C(4,6)$	2.3941	7.1409	2.9712
6	$C(4,5)$	2.3988	7.6094	2.4475
7	$C(2,6)$	2.4661	7.4140	3.1237
8	$C(1,2)$	2.4892	7.4421	3.1292
9	$C(2,1)$	2.5035	7.2952	3.2761
10	$C(2,2)$	2.5440	7.1829	3.2638
11	$C(1,3)$	2.5564	7.4461	2.9815
12	$C(4,1)$	2.6172	7.3186	3.2360
13	$C(1,4)$	2.6966	7.7193	3.0145
14	$C(2,4)$	2.7182	7.6712	3.3780
15	$C(2,3)$	2.7419	7.8359	3.1938
16	$C(3,3)$	2.8574	7.3125	3.4732
17	$C(3,1)$	2.9946	7.7103	3.2227
18	$C(1,1)$	3.0105	8.4730	3.2877
19	$C(4,4)$	3.1703	8.2811	3.6149
20	$C(4,3)$	3.4277	7.8531	3.9498
21	$C(3,2)$	3.5656	8.3227	3.8017
22	$C(4,2)$	3.7125	9.0720	4.0144
23	LEX	3.8559	9.6171	4.1472
24	$C(3,4)$	4.2008	9.9645	4.0951
25	$C(2,5)$	5.1139	10.9791	4.5531

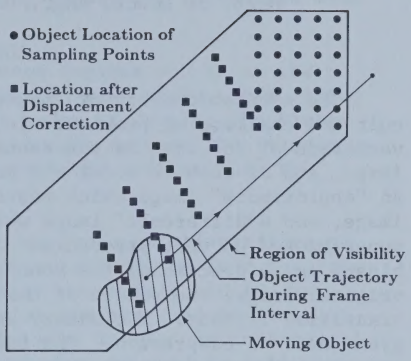


Fig. 3. Point set for lexicographic sampling of moving object.

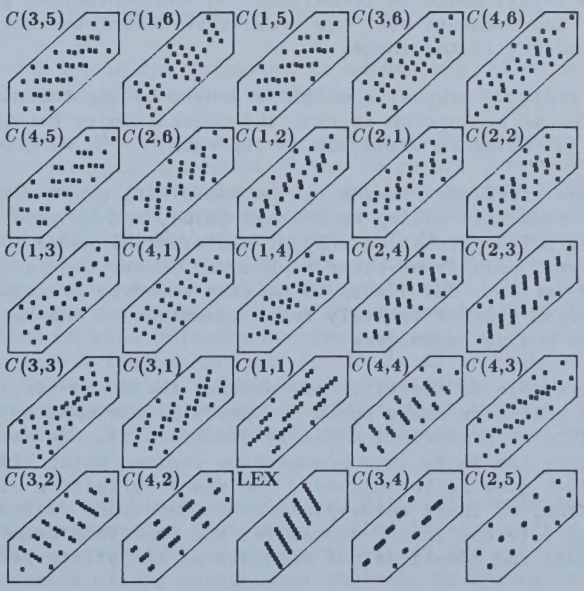


Fig. 4. Point sets for 24 congruential sampling patterns and the lexicographic sampling pattern ranked by $\lambda_{max}(\Phi)$.

AN IMPROVED APPROXIMATION IMAGE FOR MULTIPLE SOURCE CODES

T. PAPOUTSIS, L.F. CHAPARRO, AND M. KANEFSKY

Dept. of Elect. Engr., Univ. of Pittsburgh, Pittsburgh, PA

ABSTRACT

The nonstationarity of images makes their data compression a difficult and challenging problem. In this paper we consider that the space variation of the mean is the cause of the nonstationary behavior of the image, and as such we model the image as the addition of two components: an "approximate" image which represents the underlying structure of the image, and a "difference" image which corresponds to the random variations superimposed on that structure. The approximate image is obtained from a binary image describing the boundaries of the homogeneous regions of the original image, the values of their local means and interpolation in the transition regions. The binary image can be efficiently coded and with a great deal of compresion. The difference image is the result of subtracting the approximate image from the original one. It can be modeled as a stationary random field and effectively coded using predictive techniques. The procedure is completely reversible in such a way that the original image can be easily synthesized from these two components. The proposed coding procedure attempts to achieve an optimal balance between the advantages of the binary and the predictive coding procedures. An example illustrating the procedure as well as the issues involved is given at the end of the paper.

I. INTRODUCTION

The compression of images is complicated by their nonstationarity. Several solutions have been attempted with some degree of success. Composite models [1,2], adaptive techniques [3] and contouring procedures [4] provide great improvements over the predictive coding techniques that assume stationarity in the image.

In this paper we propose a multiple source coding technique for images. The basic assumption is that the nonstationary behavior of the image is due to the space variation of its mean.

Binary and stationary images can be coded with great compression, and efficiency by facsimile coding techniques [5,6], and by predictive techniques such as delta modulation (DM) or differential pulse code modulation (DPCM). However, neither of these techniques is very efficient in coding nonstationary images. Our aim is to develop a coding procedure for nonstationary images that effectively takes advantage of the facsimile and the predictive coding techniques.

If we were able to identify and subtract the space-varying mean of the image the resulting image could be approximately modeled as a stationary random field with statistics nearly Gaussian [7]. We attempt this by first segmenting the image into homogeneous regions using simple thresholding and pixel growing techniques. The exact definition of the contours of these regions, or their correspondence to important features in the image is of no interest in the procedure, and thus this segmentation step does not present the complications encountered in pattern recognitions.

The choice of threshold values will in general define transition

regions between homogeneous regions. The necessary information about these homogeneous and transition regions is given by a binary image (zeros describing the transition regions and ones the homogeneous regions) and by the values of the local averages of the homogeneous regions. The binary image can be encoded using information preserving techniques with considerable compression.

Sometimes transitions between homogeneous regions can be so sharp that there might not be a transition region. In such cases, DPCM coding would result in severe overload errors. In order to develop the binary image for those cases, our procedure automatically expands the transition regions. The decoder will reverse these expansions and get back the actual homogeneous regions.

The approximate image should characterize the nonstationarities of the image. This image should be such that when subtracted from the original image the resulting "difference" image is accurately modeled by a stationary field. We want to avoid especially the overload errors that occur at sudden nonstationary changes in the gray level. An interpolation procedure, which can be replicated in the decoder, is used to fill the gaps between homogeneous regions and to form the approximate image [8].

The difference image represents the random variations superimposed on the approximate image. Its degree of stationarity depends on the approximate image, and can be coded very efficiently using DPCM.

The proposed coding procedure makes possible the preservation of sharp features of the image and it is much simpler to implement than contour and adaptive procedures. Bilevel components of the image are reconstructed exactly, and the DPCM coding is performed without overload errors. The approximate image can in some cases be used as a good replica of the original image.

II. THE DECOMPOSITION ALGORITHM

We will now describe the procedure to encode and decode a given image. Figures 1 and 2 depict the various steps.

First we segment the image into approximately homogeneous regions. The peaks of the intensity histogram of the image are detected and used to set threshold values for segmenting the image into regions with similar pixel values. The choice of threshold values is not crucial in the process, since errors made in the segmentation will be absorbed by the difference image (to be discussed later). We calculate the local average for each of the homogeneous regions and form a table with them. The pixels with intensity values not falling within the thresholds constitute the transition regions between adjacent homogeneous regions. Isolated pixels within the homogeneous regions are set to the local average value to provide continuity in the region. Substituting the actual values of the pixels within the homogeneous regions by their local average, and setting to zero the pixels in the transition regions results in the "segmented" image.

When sharp discontinuities between two adjacent homogeneous regions occur, then the corresponding transition region is very narrow or nonexistent. In these situations, transition regions are extended by one pixel into the darker of the homogeneous regions, to insure that every homogeneous region is distinctly separated from each other. In this way,

a clear definition of the boundaries of the homogeneous regions is obtained. Making the above modifications in the transition regions of the segmented image, we obtain the "distinctly-segmented" image.

The information about the boundaries and the local averages of the homogeneous regions can be characterized by a binary image and a table of averages, respectively. Setting to unity the value of the non zero pixels in the distinctly-segmented image forms a binary image containing the boundary information (affected by the extension modifications in the transition regions).

We assume that the transition from one homogeneous region to a neighboring one is relatively smooth. This is modeled by interpolating, between the local averages of adjacent homogeneous region, in the extended transition regions. In [8] we proposed a simple interpolation algorithm. The complexity required of the interpolation algorithm depends on the desired closeness of the "approximant" image (to be developed next) to the original image, or equivalently on the degree of performance we require of the DPCM encoder when coding the "difference" image. After interpolation in the transition regions of the segmented image, we obtain the approximate image. This image may be a good replica of the original image: the flat regions of the original image are represented by the homogeneous regions, the transitions between flat regions of the original image have been smoothed out.

The difference image is the error between the approximate and the original images. It has no sharp discontinuties in its pixel values, and its statistics resemble Gaussian statistics. Because of this, the difference image can be very efficiently coded by a prediction technique such as DM or DPCM.

In summary, the encoder provides three basic components: the binary image, the table of local averages, and the difference image. It is possible to encode the binary image very efficiently by using one of the existing facsimile techniques [5,6] and achieve great compression. Not as much compression can be obtained, however, with the table of local averages. These values need to be encoded with precision in order for the decoder to be able to synthesize the image accurately.

The decoder proceeds in an inverse manner to synthesize the image. Basically, we need to form the approximate and the difference images from the three basic components. The difference image is obtained directly from the DPCM decoder. The binary image and the table of local averages permits us to form the distinctly segmented image. To get the segmented image we now need to compress the extended transition regions of the distinctly segmented image (this is the inverse of the procedure in the encoder) Performing in the transition regions of this image the same interpolation that we did in the encoder, we obtain the approximate image. This image added to the difference image gives us the reconstructed image.

III. EXAMPLE

We now apply our procedure to a 128 x 128 digitized image, which is displayed in 32 gray scale levels in Figure 3. The three-dimensional plot (Figure 4) clearly shows the sharp changes in the gray level values of the image, and especially on the regions corresponding to the "1" and the "3".

The intensity histogram of this image is shown in Figure 5. The effect of the background pixels on the histogram have been reduced for display purposes. We set thresholds on the valleys corresponding to the three peaks in the histogram, and the resulting segmented image is shown in Figure 6. It should be noticed that because of the sharp edges of the "1" and "3" no transition region exists between their adjacent homogeneous regions.

To form the distinctly segmented image (Fig. 7) we need to create transition regions around the "1" and the "3", and expand the other transition regions. This is accomplished by a simple algorithm that searches for adjacent homogeneous regions and inserts a blank pixel between them. The procedure is reversible.

Setting to unity all the pixel values in the homogeneous regions of the distinctly segmented image we obtain the binary image (Fig. 8). The facsimile coding technique presented in [6] was used to code this binary image, and we obtained a compression ratio of 0.3 bits per pixel. Better compression ratio can be obtained using more efficient procedures [5].

After interpolating [8] in the transition regions of the segmented image we obtain the approximate image displayed in Figure 9. It can be easily seen that the approximating image maintains all the main features of the original image.

Finally, we form the difference image (Figure 10) by subtracting the approximate image from the original one. The pixel values of the difference image could be negative, and for display purposes we have shifted them up. The histogram in Figure 11 is that of the difference image with the background removed. Observe that it is narrower and more nearly Gaussian than the histogram of the original image. The difference image was encoded using one-dimensional delta modulation, and a two-bit mid-treat quantizer.

We synthesized the image following the steps in Figure 2, and the result is shown in Figure 12. Note that the "1" and the "3" were exactly recovered, and that the other sharp features in the original image remained. The mean square error between the original and the reconstructed images was found to be 14.039.

For comparison, we used a one-dimensional delta modulator to encode the original image and obtained the reconstructed image shown in Figure 13. The mean square error was found to be 837.78. The maximum distortion occurred about the sharp edges of "1" and "3". Although, more sophisticated techniques can be used to get better results the sharp edges would still cause difficulties.

IV. CONCLUSIONS

The coding procedure presented in this paper uses a composite model to deal with the nonstationarity of images. The approximate image represents the underlying structure of the image, whereas the difference image relates to the random variations superimposed on that structure. Our procedure takes advantage of the high compression obtained in coding binary and approximately stationary images, and it does not require complicated calculations. The main advantage of our procedure is the preservation of sharp features.

REFERENCES

[1]. J.K. Yan, D.J. Sakrison, "Encoding of images based on a two component source model," IEEE Trans. on Communications, Vol. 25, pp. 1315-1322, Nov. 1977.

[2]. D.K. Mitrakos, and A.G. Constantinides, "Composite source coding techniques for image bandwidth compression," IEEE ICASSP-1982, pp. 1223-1226, May 1982.

[3]. W. Zschunke, "DPCM picture coding with adaptive prediction," IEEE Trans. Communications, Vol. 25, pp. 1295-1302, Nov. 1977.

[4]. M. Kunt, "Image coding" in Fundamentals of Computer Vision, O.D. Faugeras (Ed.), pp. 91-116, Cambridge University Press, 1983.

[5]. CCITT Study Group XIV, Federal Republic of Germany, "Two-Dimensional Coding Scheme," 1979.

[6]. J.L. Mitchel, "Facsimile image coding," AFIPS Conf., Arlington, VA, 1980.

[7]. B.R. Hunt and T.M. Cannon, "Nonstationary assumptions for Gaussian models of images," IEEE Trans Syst., Man and Cyber, pp. 876-882, De. 1976.

[8]. T. Papoutsis, M. Kanefsky and L.F. Chaparro, "Multiple Source Coding of Nonstationary Images," CISS Conf., Princeton, NJ, March 1984.

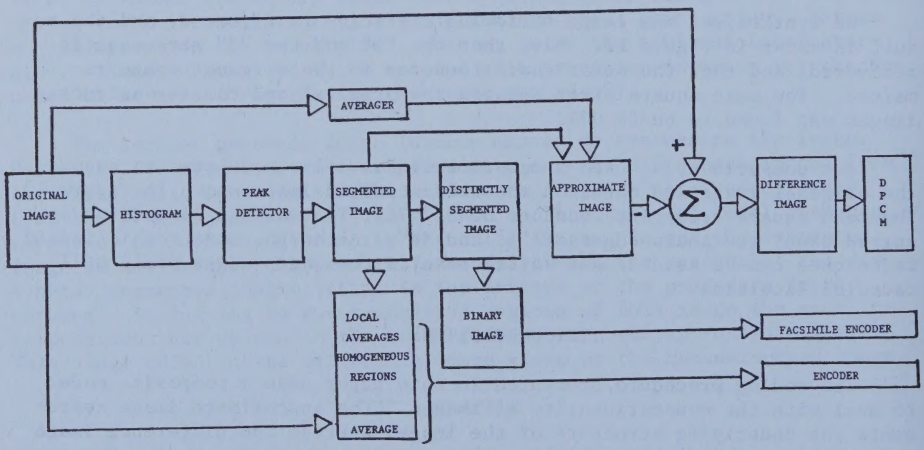


Figure 1. Encoder

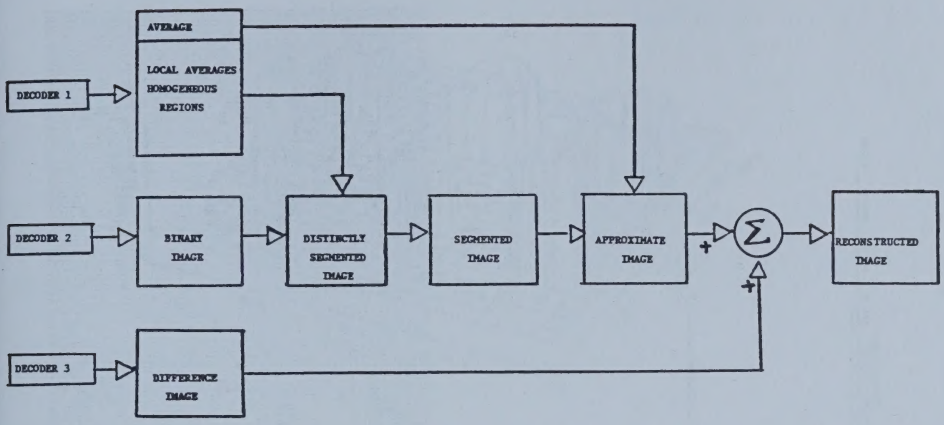


Figure 2. Decoder

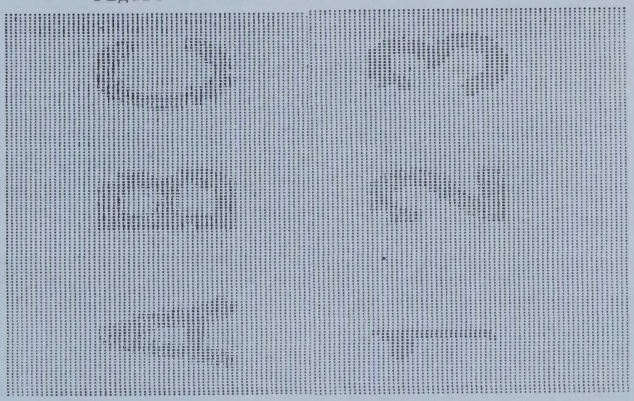


Figure 3. Original Image

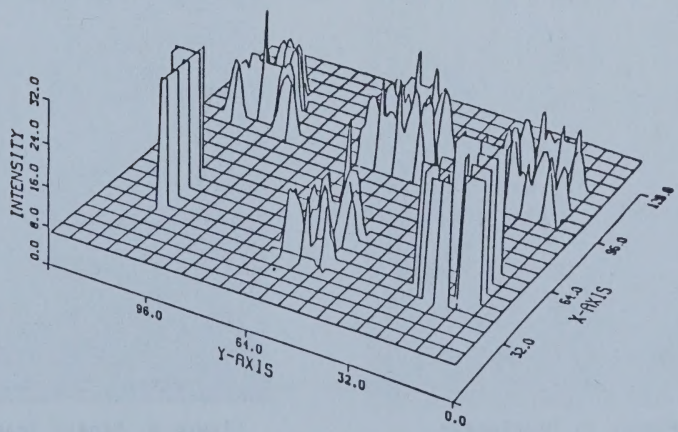


Figure 4. Three-dimensional Version

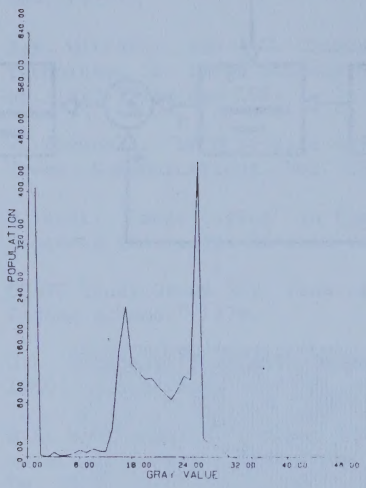


Figure 5. Histogram
(Background Reduced)

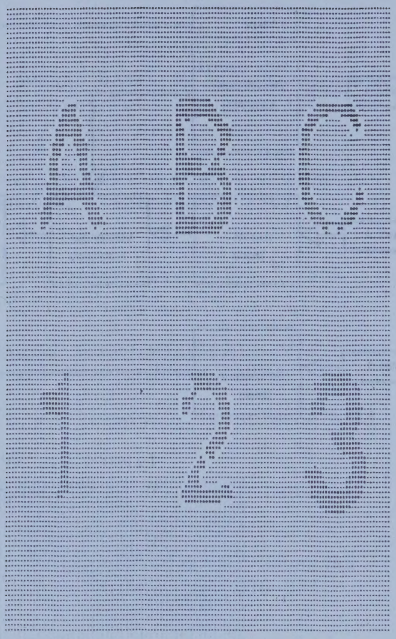


Figure 6. Segmented Image

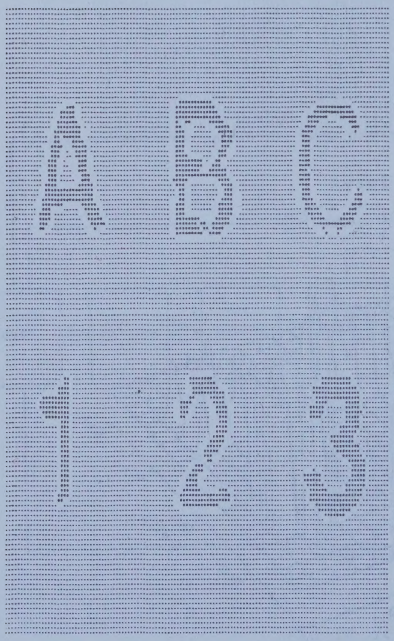


Figure 7. Distinctly
Segmented Image

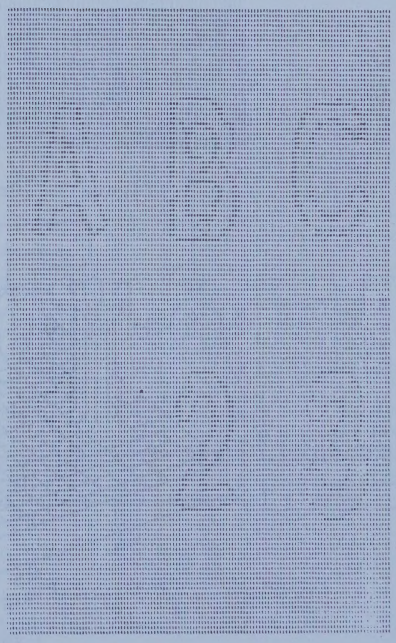


Figure 8. Binary Image

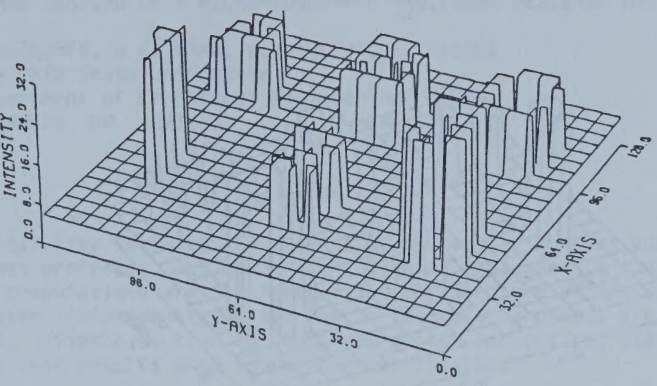


Figure 9. Approximate Image

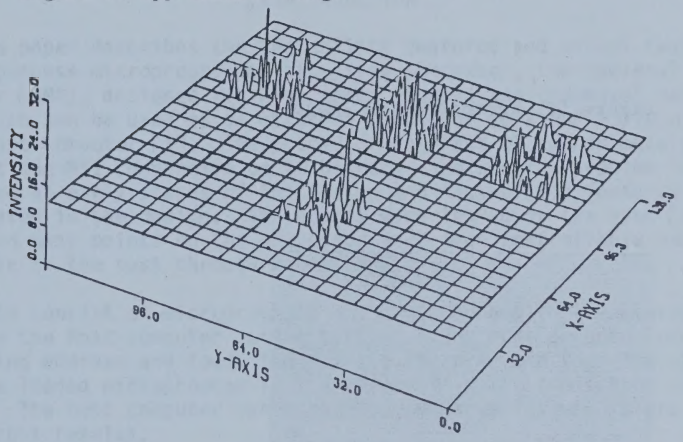


Figure 10. Difference Image

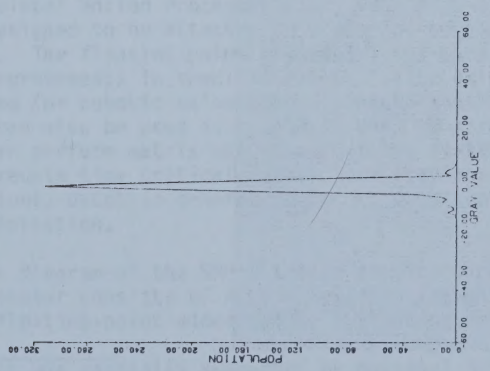


Figure 11. Histogram of Difference

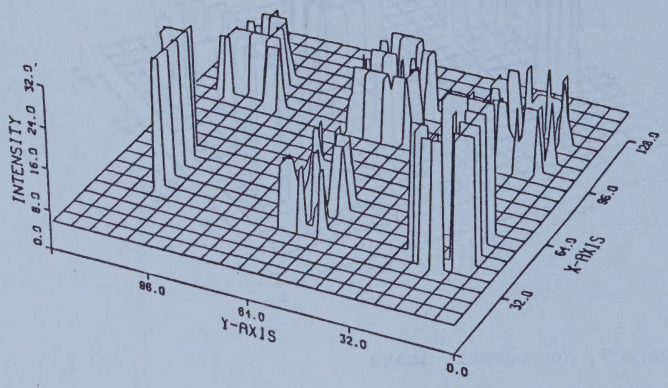


Figure 12. Reconstructed Image

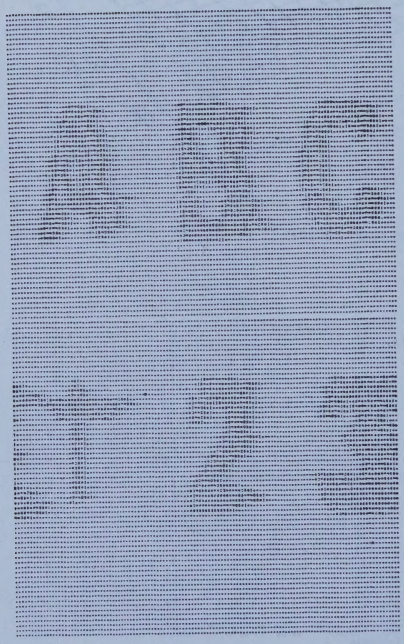


Figure 13. 1-D Delta Modulation Encoding

FUNCTIONAL TESTING OF A MICROPROGRAMMED PROCESSOR DESIGNED FOR TESTABILITY

F. OZGUNER, J.K. WANG, K.W. OLSON, E. RIBBLE
The Ohio State University
Department of Electrical Engineering
Columbus, OH 43210

ABSTRACT

Testability features and diagnostic tests of a special purpose microprogrammed processor, designed as an attached processor to speed up robotic computations are described. The architecture allows each unit to be accessed independently by the host. Test microprograms are loaded by the host. After execution the host can access various registers to compare the test results with stored correct results.

INTRODUCTION

This paper describes the testability features and actual testing of a special purpose microprogrammed bit-slice processor, the Skeletal Motion Processor (SMP), designed for high speed robotic and graphical calculations which can be used as an attached processor to a PDP 11/70 or a VAX 11/780 host computer [1]. The processor consists of approximately 2600 VLSI, LSI and MSI components on 10 boards. The testing problem was recognized at early stages of the design and testability features were incorporated in the design. The boards were tested by the host (PDP 11/70) and many points on the processor were made controllable and/or observable to the host through a bus [2].

Tests consist of microprograms which are loaded to the microcode memory by the host computer during testing. The host computer provides the loading address and loads the preload register and then the microcode RAM. The loaded microprogram is then executed while the system clock is running. The host computer can then access appropriate registers to monitor the test results.

ARCHITECTURE

The Skeletal Motion Processor (SMP) is a high-speed, microprogrammed processor designed to be attached to a general purpose host (PDP 11/70 or VAX 11/780). The floating point processor uses parallelism and pipelining for speed improvements in specific robotic calculations. Although it has been designed for robotic calculations for the synthesis of human skeletal motion, it can also be used as a general purpose array processor. The processor can perform matrix operations, solve systems of up to 128 equations and execute time critical robotic computations such as homogeneous transformations, Jacobian generation and solving system of equations using Gaussian Elimination.

A block diagram of the SMP's system architecture is shown in Figure 1. The processor consists of 4 parallel floating-point multiplier units, 4 parallel floating-point adder units, 1 floating-point sine/cosine unit,

This research was partially supported by National Science Foundation Grant No. MCS-7923670.

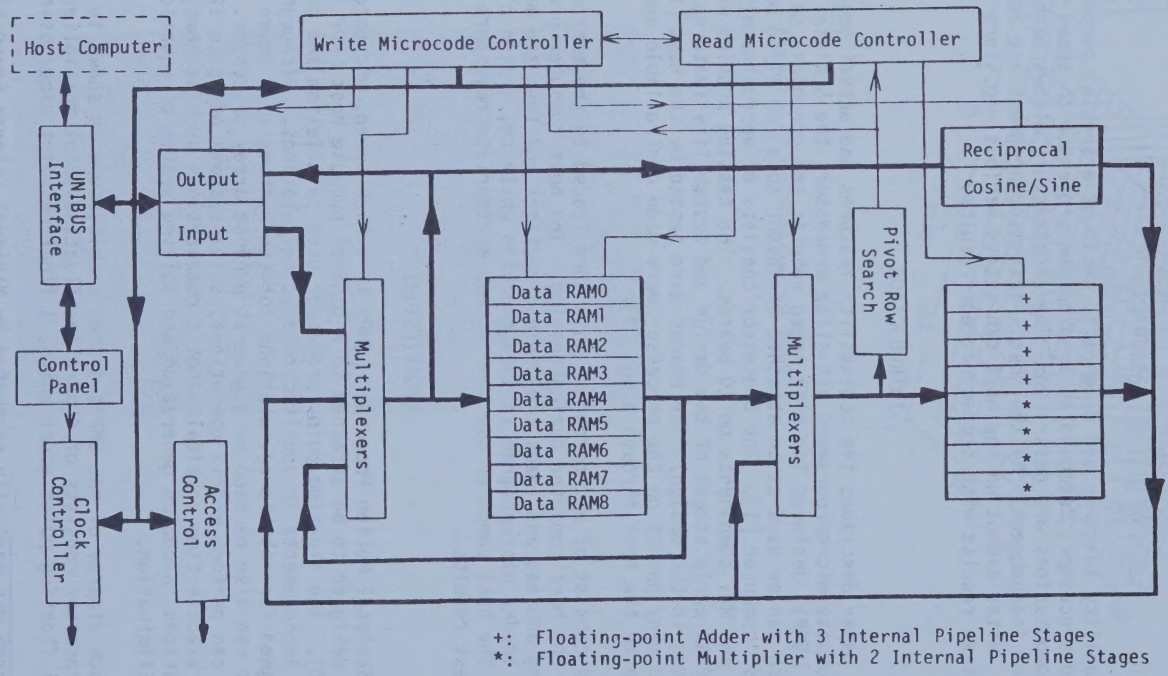


Figure 1. System Architecture

1 floating-point reciprocal unit, an input/output (I/O) interface, a pivot row search unit, 9 parallel data RAMS, and two microde controllers.

The four floating-point adders can do one of four possible operations: $\pm A \pm B$. The adder unit design consists of AMD25LS381 and AMD25LS2517 4-bit ALU slices. There is also a 32-bit combinational right shifter and a 32-bit combinational left shifter.

The 9 data memories are each 4K by 40 bits, expandable to 64K by 40, and consist of INMOS1429-55 (4K by 4) static RAMs. The RAM is configured so that they are all independently addressable. The clock period (230 ns) is partitioned such that the RAM is in the read mode the first half of the clock period and in the write mode the second half of the clock period. The read and write addresses are also completely independent for each RAM. The "read data multiplexes" supply data to the multipliers and adders and consist of sixteen sets of 4:1 multiplexers of 40 bits each. The "write data multiplexers" supply data to the RAM, the I/O, the cosine/sine unit, and the reciprocal unit and consists of four 20:1 multiplexers of 40 bits each.

The multiplier unit multiples two 32 bit mantissas and properly manipulates the sign bit and exponent. It uses three AMD29517 16-bit multipliers to multiply two 32-bit mantissas, and uses twelve AMD25LS381 4-bit ALU slices to sum the partial products.

The input section from the host computer to the processor and the output section from the processor to the host computer are each double buffered and consist of 4K by 40 bits. Communication regarding the status of the I/O buffers is most efficiently done by using interrupts.

The processor's operations are controlled by a 529-bit microcode. The "read" microcode controller (271 bits) and the "write" microcode controller (258 bits) are independent. The read controller generates "read" addresses for the 9 data RAMs, selects the adder and multiplier sources, controls the adder and multipliers, and controls pivot-row search hardware. The write microcode controller generates "write" addresses for the 9 data RAMs, selects the data to be written into the RAM, selects reciprocal, cosine/sine, and I/O sources, and controls reciprocal, cosine/sine, and I/O units. Each microcode controller uses four AMD2909A microprogram sequences, sixty-four 4-bit counters, and thirty-six 4-bit ALU slices. The controllers each have 8 loop-counters, 8 index-counters, and a 4K by 16 polygon RAM.

A multiplier (or adder) can obtain its sources from two independent RAM address and the result is stored in yet another independent address. Addresses for the nine data memories can be independently specified:

1) Directly from microcode,
2) From index counters,
3) From an index counter plus an address specified by microcode,
4) From polygon RAM,
5) From polygon RAM plus an address specified by microcode
6) From an index counter pointing to polygon RAM which contains an address.

Auto-increment of the index counters is possible. Thus many of the conventional addressing modes found in a general-purpose computer such as

the PDP 11/70 are possible. Eight loop-counters and N-way branching permit a variety of conditional branching. Pivot row hardware permits N-way branching to also be dependent on a pivot row value, which permits fast and efficient Gaussian Elimination microprograms to be written. The microcode controllers can also be instructed from the host computer by using the macro-instruction hardware, thus permitting the processor to be used as a general-purpose, attached, array processor.

Since the microprograms of the SMP are loaded from the host computer, a RAM is provided as the microstore. In addition, since one micro-instruction contains a 271-bit read microcode word and a 258-bit write microcode word and the word length of the host computer is 16 bits, a set of preload registers were designed for loading or reading the read/write microcode to or from the read/write microcode RAM.

DESIGN FOR TESTABILITY AND DIAGNOSTIC PROCEDURES

The SMP design incorporates a number of testability features. Many points on the processor are made controllable and/or observable to the host through a bus. As seen in Fig. 2 inputs and outputs of units of the processor are connected to the 16-bit HB bus through tristate buffers or registers which are enabled by the read/write access signals (RCDD, RDW1, ..., RC(15:0), WR(17:0)) generated by the Access Control Unit. The Access Control Unit communicates with the host through the UNIBUS interface and thus the read/write access signals can be sent from the host computer. To further facilitate hardware trouble shooting, a UNIBUS simulator has been designed which can be operated statically from switches. The clock controller hardware permits the processor to operate in single step and break point modes.

The microprograms required for microdiagnostics are prepared on the host computer and stored on the disk of the host computer. During testing, they are selected and loaded to the read/write microcode memory. Then, the loaded microprogram is executed to exercise the internal operation of the system while the system clock is running. The verification of test results is done by comparing the correct result with the one read back.

The Access Control Unit controls the access to different units during testing. For example, to load a microprogram into the read/write microcode RAM, the loading address is specified. Then, writing the microprogram to the read/write microcode preload register is done by WC2/WC3 and WRW1. Next, the microprogram is loaded into the read/write microcode RAM by WRW2 and WRW2 will autoincrement the loading address provided by the diagnostic/loading address counter. The diagnostic/loading address counter also provides the loading address for the read/write polygon RAM and the loading or diagnostic address for the I/O buffer RAMs as shown in Figure 2.

After being loaded, the microprogram can be executed from address 0 and in the single instruction mode. The clock controller provides special functions such as normal running mode, single instruction mode including stop clock low (SCL) and stop clock high (SCH) mode, and break-point (BP) mode for stopping clock at a specified microcode address, to synchronize the operation of various units of the SMP and to debug the hardware or software (microprograms) of the SMP. When the break-point mode is enabled for the read/write controller, the system clock will run as specified

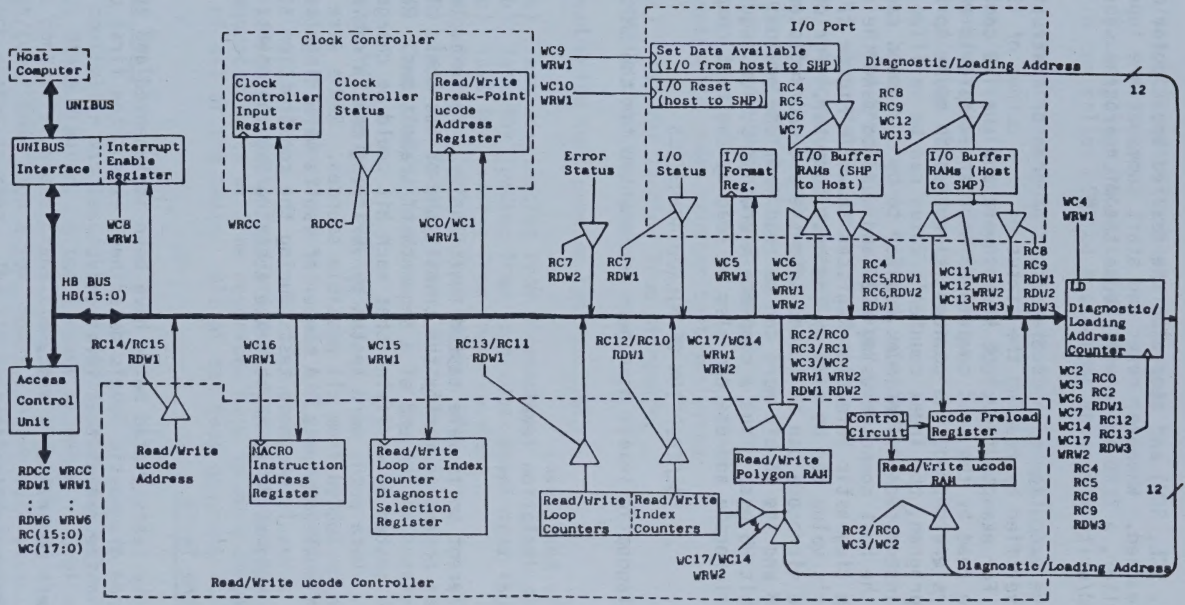


Figure 2. Observable and controllable units and bus architecture for testability.

(i.e. normal, SCL, SCH) and stop when the desired break point microcode address is reached. Whenever reset and start commands are inputted to the clock controller, a ZERO pulse will cause the microprogram sequencer to zero the read/write microcode address.

After the execution of each diagnostic program or instruction the results are verified by reading the status or output of the unit under test. For example the clock status word and the I/O controller's status can be read by the host computer. Another example is when the index counters are tested, the counter is enabled to count up as specified in the microprogram, the index counter's value can be verified after each single step instruction. The index counter to be diagnosed can be selected by the host computer using WC15 and WRW1 to send the counter number to the Diagnostic Selection Register shown in Figure 2. Then, the index counter's value can be read by the host computer. For diagnostic purposes, the microcode can be loaded into the preload register from the microcode RAM and the 16-bit word can be read from the preload register over the 16-bit HB bus. Thus a complete microinstruction requires 17 reads. The microcode address can also be read by the host computer.

DESCRIPTION OF MICRODIAGNOSTICS

Three diagnostic levels have been implemented for the SMP. These are:

1) Daily test level
2) Fault Isolation level
3) Manual test level

The different units were tested functionally. For example the clock controller was tested to verify the transitions of its state diagram by reading its status at the end of a sequence of transitions. Registers and counters were tested by verifying that each bit could be changed from 0 to 1 and 1 to 0. Data paths were tested by verifying that a register or a counter could be loaded from all possible sources. RAM's were tested using conventional RAM tests. A number of faults were diagnosed and corrected as a result of these tests during the construction stage. Tests will also be used as daily maintenance tests during the operational stage of the system.

Overview of the Tests

All of the tests listed below have been implemented and can be performed in three diagnostic levels described, except the first one which can be done in the manual test level only.

1. Access control unit test.

2. Clock controller test.

3. I/O buffer RAM test including:

    1) I/O buffer RAM 1 (host to SMP) test.
    2) I/O buffer RAM 2 (host to SMP) test.
    3) I/O buffer RAM 1 (SMP to host) test.
    4) I/O buffer RAM 2 (SMP to host) test.

4. Input/output controller test including:

   1) I/O controller (host to SMP) test.
   2) I/O controller (SMP to host) test.

5. Read/write microcode controller test including:

   1) Read/write microcode preload register test.
   2) Read/write microcode RAM test.
   3) Read/write microprogram sequencer test including:

      (a) Jump microcode D-field, microprogram counter and true/false multiplexer test.
      (b) Jump MACRO test.
      (c) Address holding register test.
      (d) Push/pop stack test.
      (e) OR-field test
      (f) Conditional branch test.

   4) Read/write index counter test including:

      (a) Test for loading from microcode.
      (b) Up counting test.

   5) Read/write loop counter test including:

      (a) Test for loading from microcode.
      (b) Test for loading from loop counters.
      (c) Test for loading from index counters.
      (d) Test for loading from polygon RAM.
      (e) Test for loading from I/O counters.
      (f) Down counting test.

   6) Read/write polygon RAM test.

   7) Test for arithmetic units, data RAM and data paths.

The clock controller test, read/write microcode preload test and the microprogram sequencer tests will be described here in more detail as representative of all of the tests. All of the test programs were written in PASCAL.

Clock Controller Test

The clock controller was tested by verifying the state transition sequences in its state diagram. After a reset sequence is applied to reset the clock controller state transitions are initiated to check the "stop clock high/low," and "high & low" modes. After each transition sequence the clock status is read. The I/O controller was tested in a similar way. During construction, the clock controller test detected a design error where the clock controller was making a wrong transition.

Read/Write Microcode Preload Register Test

The preload registers are tested by shifting the following 18 tests into the 16 bit preload registers shown in Figure 3.

Data 1 = 0000 0000 0000 0001	Data 10 = 0000 0010 0000 0000
Data 2 = 0000 0000 0000 0010	Data 11 = 0000 0100 0000 0000
Data 3 = 0000 0000 0000 0100	Data 12 = 0000 1000 0000 0000
Data 4 = 0000 0000 0000 1000	Data 13 = 0001 0000 0000 0000
Data 5 = 0000 0000 0001 0000	Data 14 = 0010 0000 0000 0000
Data 6 = 0000 0000 0010 0000	Data 15 = 0100 0000 0000 0000
Data 7 = 0000 0000 0100 0000	Data 16 = 1000 0000 0000 0000
Data 8 = 0000 0000 1000 0000	Data 17 = 0000 0000 0000 0001
Data 9 = 0000 0001 0000 0000	Data 18 = 0000 0000 0000 0010

The patterns that are shifted out are read by the host computer and compared with the stored correct patterns. The patterns test for 0 to 1 and 1 to 0 transition of all bits. The position of the error can be detected. However the register that caused the error cannot be located.

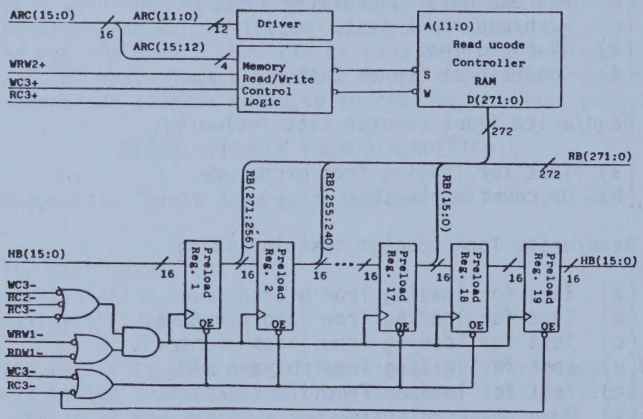


Figure 3. Read Controller Microcode Preload Register and Microcode RAM

Microprogram Sequencer Tests

Jump Microcode D-Field Test. This test checks if the microprogram sequences can jump to the address specified in the microcode D-Field. The microprogram is given below:

Address	Description
0	unconditionally true, jump microcode D-Field = 1
1	unconditionally false, jump microcode D-Field = 2
2	unconditionally true, jump micorcode D-Field = 4
4	unconditionally false, jump microcode D-Field = 8
8	unconditionally true, jump microcode D-Field = 16
16	unconditionally false, jump microcode D-Field = 32
32	unconditionally true, jump microcode D-Field = 64
64	unconditionally false, jump microcode D-Field = 128
128	unconditionally true, jump microcode D-Field = 256
256	unconditionally false, jump microcode D-Field = 512
512	unconditionally true, jump microcode D-Field = 1024
1024	unconditionally false, jump microcode D-Field = 2048
2048	unconditionally true, jump microcode D-Field = 4095
4095	unconditionally false, jump microcode D-Field = 4095

The next microinstruction address should equal to the D-Field at the end of each microinstruction execution. The true/false multiplexer is also tested.

Microprogram Counter Test. This test ensures that the microprogram sequencer can properly increment the microcode address from 0 to 4095. It would require 4096 microinstructions to do the complete test; however, the test implemented only contains 77 microinstructions as each hexadecimal digit is tested by counting from 0 to F. The microprogram is given below:

```
      Address                          Description
------------------      ------------------------------------------
 0=0000 0000 0000       continue            test first
 1=0000 0000 0001       continue            slice

 15=0000 0000 1111      continue
 16-0000 0001 0000      jump D=31
 31=0000 0001 1111      continue
 32=0000 0010 0000      jump 3=47
 47=0000 0010 1111      continue
 48=0000 0011 0000      jump D=63

  :                     : (alternate the continue and jump
  :                     :  to test the second slice)
 239=0000 1110 1111     continue
 240=0000 1111 0000     jump D=255
 255=0000 1111 1111     continue
 256-0001 0000 0000     jump D=511
 511=0001 1111 1111     continue
 512=0010 0000 0000     jump D=767
 767-0010 1111 1111     continue
 768=0011 0000 0000     jump D-1023

   :                    : (alternate the continue and jump
   :                    :  to test the third slice)

3839=1110 1111 1111     continue
3840=1111 0000 0000     jump D=4095
4095=1111 1111 1111     jump D=4095
```

OR Field from Loop Counters Test. The OR-Field from the loop counters test ensures that the microcode address can be properly ORed with the output of the loop counters. Since the outputs of the loop counters are tied together, it is sufficient to use only loop counter 0 to test the OR-Field. During execution, each microprogram loads loop counter 0 with propagating 1; then, the microprogram directs the microcode address to jump to address 0, but the microcode address is ORed with the output of loop counter 0. The microprograms are shown below.

```
 Address                          Description
----------      ------------------------------------------------------
Set  0:  0      Load loop counter 0 with 0000 0000 0001, continue
         1      Jump 0, but ORed with loop counter 0's value.

Set  1:  0      Load loop counter 0 with 0000 0000 0010, continue
         1      Jump 0, but ORed with loop counter 0's value.

     :                          :
```

Address		Description
:		(Each time the set number is increased, a word with a 1
:		propagated left is loaded to loop counter 0 at address 0.
:		The instruction at address 1 is unchanged.)
:		:
Set 11:	0	Load loop counter 0 with 1000 0000 0000, continue
	1	Jump 0, but ORed with loop counter 0's value.
Set 12:	0	Load loop counter 0 with 1111 1111 1111, continue
	1	Jump 0, but ORed with loop counter 0's value.

After each microprogram is loaded, the system clock is started in the stop clock low mode and toggled twice. Then, the verification is done by comparing the current read/write microcode address with the correct one.

The RAM's were tested by standard memory tests. The stack which consists of four 12-bit locations was tested by pushing data four times and popping five times so that the location of the first pop was addressed by the stack pointer. This was repeated five times so that all bits could be tested by the propagating 1's pattern. Conditional branch hardware was tested by generating the different conditions and running microprograms to exercise different branch operations and verifying microcode addresses.

Since all components of the system, except for the arithmetic units, had been tested, it was possible to use the SMP as an attached processor and test the arithmetic units functionally with application software. The adders and multipliers were tested by "racing ones" across the mantissa and exponent of both input sources, and the host computer observed the output results. The reciprocal unit was also tested using the "racing ones" as input. The cosine/sine unit was tested with two kinds of tests: 1) "racing ones" test to the mantissa and exponent, and 2) random-number generator test that generated real numbers for inputs to the unit and the host computer observed the output results.

CONCLUSION

During the construction of SMP boards a number of errors were detected by the tests. Among the faults detected were a short between two RAM address lines, a faulty RAM chip and incorrect stack operation because of incorrect wiring of a pin.

REFERENCES

[1] Olson, K.W. and E.A. Ribble, "Skeletal Motion Processor for High Speed Robotics and Graphical Computations," Proceedings of the IEEE Inter. Conference on Robotics, 1984.

[2] Wang, J.K., Test Generation and Diagnostics for the OSU Skeletal Motion Processor, M.S. Thesis, The Ohio State University, Columbus, Ohio, August 1983.

TESTABLE DESIGN OF SEQUENTIAL MACHINES USING COMPACTION FUNCTIONS AND CHECKING EXPERIMENT

Kewal K. Saluja,
Department of Electrical and Computer Engineering,
University of Newcastle,
N.S.W., 2308, Australia.

R. Dandapani,
Department of Mathematics and Computer Science,
Youngstown State University,
Youngstown, Ohio, 44505, U.S.A.

ABSTRACT

The problem of testing sequential machines using checking experiments and different compaction functions is investigated. Procedures are given for augmenting a single output sequential machine by adding extra inputs to make it testable. The augmented machine can be determined to be faulty or fault free by making only a single observation at the end of the checking experiment.

I INTRODUCTION

The explosive growth in the size of digital circuits on a chip has compounded the problem of testing these circuits for correct operation [1]. A digital circuit can be tested for faults by separately testing the combinational and sequential portions. It may be necessary to add extra inputs/outputs to effect the separation for testing. A number of modern methods [2]-[6] are available for testing combinational circuits. These methods can be classified according to the input generators used in testing. They are: (i) Exhaustive, if all circuit inputs are generated; (ii) Random, if a subset of the set of all possible inputs are generated; (iii) Pre-selected, if a set of test inputs derived by conventional test generation methods using a specific fault model is generated at time of testing. Most of the methods perform fault detection using data compaction at the output of circuit. Sequential circuits can be tested using checking experiments consisting of distinguishing, synchronising, and transfer sequences [7]; Pre-selected test sequences (generated with a specific fault model) can also be used in testing for specific faults.

Alternatively, digital circuits can be designed with testability considerations. Partitioning [1] is a method proposed for combinational circuits. Level Sensitive Scan Design (LSSD) [8] provides a uniform approach to separating the combinational and sequential portions of a circuit. The combinational circuits can then be tested using the registers in the sequential portion.

Recently, methods mave been proposed to modify sequential machines (state tables) by adding a controllable input [9-12]. These methods use the extra input to order the sequences in a checking experiment. Application of the ordered input sequences results in outputs that can be compacted. This reduces the amount of data to be observed.

In this paper we propose a method of modifying sequential machines by adding a controllable input to simplify testing by checking experiments. We will show that it is sufficient to observe one output value to determine whether the machine is faulty or fault-free. The method presented lends itself easily to designing circuits for these machines. Background material is presented in Section II. Definitions and notations are developed in Section III. Formal results are given in Sections IV and concluding remarks

are contained in Section V.

II BACKGROUND

The use of checking experiments to verify proper functioning of a sequential machine was introduced by Moore [13] and Hennie [7]. The experiments comprise of the following input sequences:

(i) Synchronising sequences to arrive at a given state from an unknown state.

(ii) Distinguishing sequence (DS) to verify the existence and identification of all states.

(iii) Transfer sequences to verify the transitions from one state to another.

These sequences are collected to form a partially ordered set (Test Set) of input sequences for testing. The procedure is to apply a sequence from the test set and observe the output sequence produced by the machine. Any departure from the reference output sequence indicates improper functioning of the machine. The procedure is repeated for all sequences in the Test Set.

Such procedures require the existence of DS for testing, but there are machines which do not have a DS. It was shown by Hennie [7] that the existence of DS simplifies testing. The length of a DS in the worst case will be an exponential function of the number of states in the machine.

In testing using these conventional procedures it is necessary to observe outputs produced by each input in the DS. Also, the reference output sequence produced by the application of DS need to be stored, for each state of the machine. This may require a prohibitively large amount of storage and testing time. Reduction in number of observations and storage can be achieved by data compaction at the output as shown in Figure 1. In this scheme, the outputs are collected for a predetermined period of time and compressed by the Data Compactor (DC) as a 'signature'. The signature is then compared with a reference signature to determine if the circuit was faulty. A multiplicity of signatures will be produced during the course of testing. The number of signatures produced depends on the time instants chosen to read-out the compacted output and is determined by the Test Set along with the DC chosen.

For single-output sequential circuits Venkatraman and Saluja [10] use Transition Counter (TC) as DC after modifying the circuit design with a controllable input. Number of states remain the same as in the original circuit. Let n and m_i be the number of states and number of inputs in the given (original) machine, respectively. Then,

$$\text{Test Length}^{*} = 4n^2 + \frac{3}{2} m_i n^2 + \frac{1}{2} m_i n + n$$

The Test Set includes sequences for which it is not necessary to observe all outputs even in conventional testing. The sufficient number of observed outputs can be computed as:

---

* This equation can be derived from Venkatraman et al [10]. In both [10] and [12] the test length bound is stated at $\frac{5}{2}n^2 + \frac{3}{2} m_i n^2$ where m_i includes the extra controllable input.

$$\text{Observed outputs}^{**} = 3n + m_i n + m_i n + n$$

Using data compaction only the signatures need to be observed. It can be computed as:

$$\text{Observed signatures}^{**} = 2\, m_i n + 4n$$

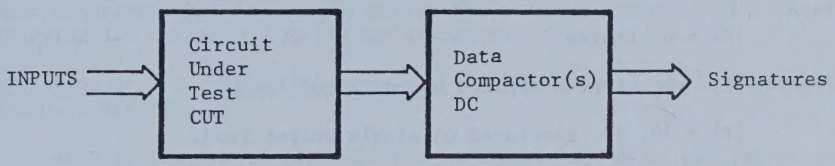


Figure 1: Testing Using Data Compactors

Hassan [12] uses Linear Feedback Shift Register (LFSR) as DC after modifying the circuit design in a manner similar to Venkatraman and Saluja [10]. In this case,

$$\text{Test length} = (4n^2 + \tfrac{3}{2} m_i n^2 + \tfrac{1}{2} m_i n + n) + (m_i n + 3n + 1)$$

$$\text{Observed outputs} = (3n^2 + m_i n^2 + m_i n + n) + (m_i n + 3n)$$

$$\text{Observed signatures} = 2\, m_i n + 4n$$

The size of test length and observed outputs are increased as the DS required is one bit longer than the DS used in [10].

For a 4 state ($n = 4$), 2 input ($m_i = 2$) machine these values are:

Test/Data Compactor	TC	LFSR
Test length	120	141
Observed outputs	92	112
Observed signatures	32	32

Thus, instead of 92 output observations only 32 are needed with TC as data compactor. The test length remains the same but overall testing time is reduced due to the fact that read-outs are performed at external clock speeds which are normally an order of magnitude slower than the internal clock.

III DEFINITIONS AND NOTATION

The sequential machines considered in this paper are finite state, synchronous, deterministic and of Mealy type with a single output only. The results can be applied to Moore type machines also.

** These equations include the observations for output produced by application of Synchronizing Sequence following some application of Distinguishing Sequence. We will show in Section IV that these observations are not necessary.

Definition 1: A machine (M) is a quintuple,

$$M = (S, I, Y, \delta, \lambda)$$

where S, I, Y are set of states, inputs and outputs, respectively. δ will denote the next state function and λ will be the output function.

If $|X|$ denotes the cardinality of set X then

$|S| = n$ states

$|I| = m_i$ inputs applied on m input leads

$|Y| = \{0, 1\}$ generated on single output lead.

Note that m_i represents the size of input alphabet. The number of input leads or connections is m.

δ: $S \times I \rightarrow S$ and λ: $S \times I \rightarrow Y$ are onto functions.

Definition 2: A Distinguishing Sequence (DS) is a sequence of inputs from I which produces a distinct sequence of outputs from Y when the machine M is started in each of the states in S.

M may have any number of DSs or may not have a DS. The start state of M can be determined by applying a DS and observing the output sequence.

The set of output sequences that a DS can produce can be transformed by a function for ease of identification. If an output transformation function is fixed then a DS needs to be selected whose set of possible output sequences belong to the input domain of the function. In general all DS may not have this property. If there is a DS for a function F then we will call it FDS.

Definition 3: An F-Distinguishing Sequence (FDS) is a DS whose output transformation function is F.

Transition Count Distinguishing Sequence defined in Venkatraman [10] is an FDS where F is the transition count function. Such a function is easy to implement for binary transition counting. A second example is the Signature Distinguishing Sequence in Hassan [12]. Here the function F can be implemented by a primitive polynomial based Linear Feedback Shift Register. Other functions, e.g. Syndrome, Walsh functions, etc., can also be defined.

Definition 4: A Synchronizing Sequence (SS) is an input sequence which leaves the machine in a specified final state regardless of the start state.

Notation: $T(s_i, s_j)$ will mean an input sequence which takes the machine from state s_i to s_j. Such a sequence is called Transfer Sequence (TS).

A machine can be verified for correct operation by applying input sequences as follows and observing the corresponding output sequences:

(i) Take M to a known state, say s - apply SS.

(ii) Check for existence of all n states - apply n ordered tripled (TS, DS, SS) with TS effecting the transfer from s to a distinct state with each triple. The n triples can be applied in any order. The last triple will

leave M in s.

(iii) Check for all possible transitions between states - apply $m_i n$ ordered quadruples (TS, an input from I, DS, SS), with TS effecting the transfer from s to a distinct state, and from each state m_i transitions are checked, one for each input alphabet. The $m_i n$ quadruples can be applied in any order.

Definition 5: A Checking Experiment (CE) is a partially ordered set of input sequences that is used to verify the correct operation of a machine by observing its corresponding output sequences.

Definition 6: An F-Checking Experiment (F-CE) is a CE whose distinguishing sequences are FDSs.

Let Z be the set of non-negative integers, and $\mathbb{R}$ be the set of rationals. For a binary sequence $R = r_1 r_2 \ldots r_p$ denote, $Y^x = \{R \mid p = x\}$, the set of all sequences from Y of length x.

The three functions which shall be considered in this paper are:

(1) TC: $Y^n \to Z$, where $TC(R) = \sum_{i=1}^{n-1} r_i \oplus r_{i+1}$ when $\oplus$ is Exclusive-OR.

(2) LFSR: $Y^{n+1} \to Y^t$, where $t = \lceil \log_2 n \rceil$ (ceiling of $\log_2 n$) and

$$LFSR(R) = \begin{cases} \text{remainder of dividing the polynomial represented} \\ \text{by R by a primitive polynomial of degree } t \end{cases}$$

(3) SYN: $Y^n \to \mathbb{R}$, where $SYN(R) = \frac{1}{n} \sum_{i=1}^{n} r_i$

SYN was used in Savir [14].

IV SINGLE-OUTPUT MACHINES

We now give an algorithm to modify a single output sequential machine M to M^* using an extra input so that M^* is easily testable. The procedure used is similar to Algorithm 2 in Venkatraman [10] and Augmentation Algorithm A1 in Hassan [12].

Algorithm 1: Augment M to M^*.

Step 1: Select states s_i and s_j such that s_i is an immediate successor of s_j. Number the state s_i as n-1 and s_j as 0, and the remaining n-2 states from n-2 to 1, arbitrarily.

Step 2: Assign the next state functions for the extra input ε as follows:

$$\delta(i,\varepsilon) = i-1 \quad \text{for } i > 0 \text{ and } 0 \text{ for } i = 0.$$

Step 3: For the output function choose one of the following three functions:

(i) $\lambda_1(i,\varepsilon) = i \bmod 2$; (ii) $\lambda_2(i,\varepsilon) = 0$ for $i = 0$ and 1 otherwise;
(iii) $\lambda_3(i,\varepsilon)$ = same as λ_2 above. □

Notation: By ε_i we will mean a sequence of k ε's.

Lemma 1: The sequential machine M^* obtained from M by applying Algorithm 1 has an FDS.

Proof: F as TC and LFSR for λ_1 and λ_2 respectively, the lemma is proved in [10] and [12]. For λ_3 and F as SYN it is a simple matter to show that SYN value for this sequence is i/n. Thus for any two distinct states, i,j $SYN(\lambda_3(i,\varepsilon_n)) \neq SYN(\lambda_3(j,\varepsilon_n))$. ▽▽▽

Lemma 2 [10]: ε_n is SS for machine M^* obtained by applying Algorithm to M.

For the three functions, TC, LFSR and SYN, an FDS is also an SS. The final state in these cases is always state 0. So, it is not necessary to apply SS after FDS in the checking experiments.

An F-CE for machine M^* consists of an initializing sequence SS followed by ordered pairs and triples of the form: (TS, FDS) and (TS, an input from I, FDS). These n pairs and $n.m_i$ triples can be applied in any order. Therefore, there are $(n+n.m_i)!$† possible F-CE's given an FDS.

In Section II the number of observed signatures was shown to be less than the number of observed outputs. We will now show that by ordering the pairs and triples in an F-CE we can reduce the number of observed values to one.

The approach is to order pairs and triples so that the signatures resulting after application of these form an ordered sequence. For example, if TC-DS is used then pairs can be ordered so that the resulting signatures are in ascending TC values. The correctness of these signatures can then be verified by a circuit which identifies any departure from the ascending sequence [15]. Thus it is sufficient to observe only the final result of applying a series of sequences.

We will first give an example of the procedure and then formalize it.

Example: Consider the machine M given in Table 1.

Table 1: Machine M

State\Input	0	1
s_1	$s_1/_1$	$s_4/_0$
s_2	$s_1/_1$	$s_3/_0$
s_3	$s_3/_1$	$s_2/_1$
s_4	$s_2/_1$	$s_3/_1$

Table 2: M^* of M

State\Input	0	1	ε
3	2/1	1/0	2/1
2	2/1	0/0	1/0
1	1/1	3/1	0/1
0	3/1	1/1	0/0

Apply Algorithm 1 with λ_1 as output function to machine M:

$$\delta(s_4, 0) = s_2.$$

Number s_4 as 0 and s_2 as 3. Arbitrarily number s_3 as 1 and s_1 as 2.

The transitions and outputs under input ε are easily obtained by using functions δ and λ_1. The resulting machine M^* is given in Table 2.

By Lemmas 1 and 2, $\varepsilon_4 = \varepsilon\varepsilon\varepsilon$ is both FDS and SS for M^* with F = TC. Then TC-CE for M^* is the partially ordered set,

$$\text{TC-CE} = \{\varepsilon_4, (T(0,i),\varepsilon_4)_{i=0,1,2,3}, \{(T(0,i),I_j,\varepsilon_4)_{I_j=0,1,\varepsilon}\ i=0,1,2,3\}$$

There are 4 pairs and 4.3 = 12 triples of sequences in the experiment which can be applied in any one of 16! different ways. We will show a method of ordering the sequences.

The pairs $(T(0,i),\varepsilon_4)_{i=0,1,2,3}$ are ordered in ascending values of i. To order the triples we construct Predecessor-Successor Tables PST-0 and PST-1, Tables 3 and 4 respectively, by the method given below:

† x! indicates factorial x.

Table 3: PST-0

PdS\PS	3	2	1	0
3			1	
2			ε	1
1				
0				ε

Table 4: PST-1

Pds\PS	3	2	1	0
3		0,ε		
2		0		
1	1		0	ε
0	0		1	

An entry in row i (Predecessor State denotes as PdS in the tables), column j (Present State denoted as PS in the tables) of PST-0 (PST-1) indicates the input needed to take machine M^* from state i to state j with output 0(1). These tables can be easily generated from Table 2.

Column 0 of PST-0 is first picked. Entries in rows 0 and 2 indicate that the triples selected first are $(T(0,0),\varepsilon,\varepsilon_4)$ and $(T(0,2),1,\varepsilon_4)$. Column 0 of PST-1 is then picked. Entry in row 1 means we select $(T(0,1),\varepsilon,\varepsilon_4)$ as the next triple.

The above procedure of picking a column from PST-0 and PST-1 is repeated for all states, in order.

The final ordered sequences are shown in Table 5.

Table 5: An 0-TC-CE for M^* of Table 2

Sequence Number	Input Sequences			Observed Output Sequences									Remarks
				Conventional			TC-CE			0-TC-CE			
	1	2	3	4	5	6	7	8	9	10	11	12	13
1			ε_4									γ	Get M to state 0 by applying SS
2	T(0,0)	ε_4			0000			0				α	Check for DS. Each DS is also SS. So, M is left in state 0 after testing each state
3	T(0,1)	ε_4			1000			1				α	
4	T(0,2)	ε_4			0100			2				α	
5	T(0,3)	ε_4			1010			3				γ	
6	T(0,0)	ε	ε_4		0	0000		0	0				Checks for transitions into state 0. Transitions sorted by output.
7	T(0,2)	1	ε_4		0	0000		0	0		β		
8	T(0,1)	ε	ε_4		1	0000		1	0		β	α	
9	T(0,2)	ε	ε_4		0	1000		0	1				Check for transition into state 1.
10	T(0,3)	1	ε_4		0	1000		0	1		β		
11	T(0,0)	1	ε_4		1	1000		1	1				Transitions sorted by output.
12	T(0,1)	0	ε_4		1	1000		1	1				
13	T(0,2)	0	ε_4		1	0100		1	2				Check for transitions into state 2. There are no transitions with output 0.
14	T(0,3)	0	ε_4		1	0100		1	2				
15	T(0,3)	ε	ε_4		1	0100		1	2			α	
16	T(0,0)	0	ε_4		1	1010		1	3				Check for transitions into state 3. There are no transitions with output 0.
17	T(0,1)	1	ε_4		1	1010		1	3		(@)	(@)	
Total observations made				0	28	48	0	16	12	0	1	1	* α and β can be OR-ed to reduce number of observations to 1.
					76			28			2*		

Outputs left blank in the table mean they are not observed for testing. If conventional testing is performed then 76 outputs are observed while 28 observations are needed if transition counter (TC-CE) is employed. (These numbers are less than the numbers given in Section II, as we no longer use SS following the application of DS). Due to our rearrangement, these 28 observations have been given a structure indicated by ordered - TC-CE (0-TC-CE). The block diagram of Figure 2 shows a method of recognizing the 0-TC-CE observations. The procedure used is given below for which F is TC

and Reference Register (RR) is assumed to be a counter of same size as TC.

Step 1: M^* is driven to state 0 by applying SS.

Step 2: RR is set to 0. Sequences 2 to 5 are applied in succession. RR is incremented by 1 at the end of each sequence. After each sequence is applied the TC in F register and RR are compared. If they are unequal, the α-latch is set to 1, otherwise it is left unchanged. The final value (0 or 1) shown in row 5, column 11, is left in α-latch. If α is 1 then M^* is faulty and the value will not change for the remainder of the experiment.

Step 3: RR is set to 0 and the Reference Value (RV) in Output Verifier (OV) is set to 0. Sequences 6,7,8 (indicated by a solid line after row 8) are applied in succession. After each sequence is applied the comparators either set the corresponding latches α and β to 1 or leave them unchanged. At the conclusion of sequence 7 (indicated by a broken line after row 7) RV is toggled to 1. At the conclusion of sequence 8 the RV is toggled back to 0 and RR is incremented by 1.

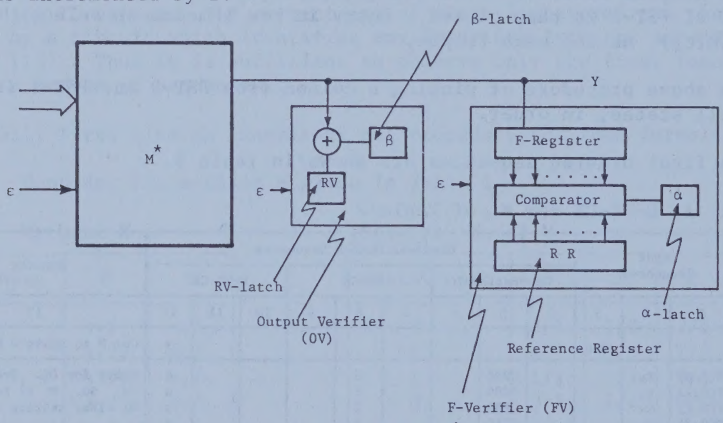


Figure 2: Structure for Testing M^*

Step 3 is repeated three more times as shown in Table 5. The γ entry indicates where RR is reset to 0 and α entries indicate where RR is incremented by 1. β entries mean RV is toggled. The only observation needed are the circled α and β values.†

If the value in either α or β latch is 1 at the conclusion of experiment then M^* is faulty. Thus it is sufficient to observe only logical OR of α and β latches at the conclusion of the experiment.

This concludes the example.

We now formalize the procedure.

Algorithm 2: Rearrange an F-CE to form O-F-CE to automate output verification.

Step 1: Apply Algorithm 1 to M with λ_i(i=1,2 or 3) as output function. Let M^* be the resulting machine, then

$$\text{F-CE} = \{SS,(T(0,i),FDS)_{i=0,1,\ldots,n-1}\{(T(0,i),I_j,FDS)_{j=0,1,\varepsilon}\}_{i=0,\ldots,n-1}\}$$

where $SS = \varepsilon_n$; $FDS = \varepsilon_n$ or ε_{n+1} depending on F.

---

† The TC register is set to 0 prior to application of each sequence.

Step 2: Construct Predecessor/Successor tables with outputs 0 and 1 (PST-0 and PST-1. Each of these two tables contain n rows and n columns numbered 0 to n-1. Entries in the table are from $\{I\} \cup \{\varepsilon\}$ (inputs to M^*). An entry in row i, column j of table PST-0 (PST-1) indicates the input that will take M^* from state i to state j with output 0(1).

Step 3: Set O-F-CE to the sequence ε_n.

Step 4: Add the pairs of sequences, (T(0,i), FDS) for i = 0,1,...,n-1, in order, to O-F-CE.

Step 5: Repeat the following two substeps for i = 0,1,..., n-1 alternately.

Step 5a: If column i of PST-0 has x in row j, then add the triple (T(0,j),x,FDS) to O-F-CE. Repeat for all rows. The order in which rows are picked is not important.

Step 5b: Same as Step 5a with PST-1. □

Theorem 1: Any single output machine can be augmented by using an extra input in such a way that the augmented machine can be determined to be faulty or fault-free by observing only one value at the end of a checking experiment.

A formal Proof of the theorem is not included here for space limitation, but can be found in [17].

V DISCUSSION AND CONCLUSIONS

In this paper we studied testing of sequential machines using checking experiments. We have presented an algorithm to augment machines by adding an extra input(s) to make them testable. The algorithm is a systematic way of rearranging the states in the state table description of M and to add the transitions and outputs for the controllable input . This lends itself naturally to the design of machines which are easily testable. Such an approach provides an alternative to design techniques such as scan methods [1] and will be presented in a later report.

Augmented machine M^* is tested by the ordered checking experiment developed in Algorithm 2, using the block diagram of Figure 2. Theorem 1 guarantees that M^* can be tested by making only one observation at the conclusion of the checking experiment O-F-CE. The compression functions we have considered are TC, LFSR and SYN. In VLSI environment such functions can generally be expected to be available on chip along with M^*. These functions can be used while testing M*, thus reducing the need for extra hardware to implement these functions.

As only one observation is required to test M^*, it may be necessary to make the test circuit self-testing [16]. Alternatively, tests can be developed for testing the test circuit. The simplicity of the test circuit makes both these approaches equally attractive.

ACKNOWLEDGEMENT

The authors are thankful to Professor C.R. Kime of the University of Wisconsin, Madison for many helpful discussions.

The work reported here was in part supported by the Department of Science and Technology and ARGC Australia. This work was carried out while R. Dandapani was vising the University of Newcastle, N.S.W., Australia.

REFERENCES

[1] T.W. Williams and K.P. Parker, "Design for Testability - A Survey", *IEEE Trans. Comp.*, Vol. C-31, No. 1, Jan. 1982, pp. 2-15.

[2] E.J. McCluskey, "Built-in Verification Test", *Digest 1982 Int. Test Conf.*, Philadelphia, Oct. 1982, pp. 132-190.

[3] J. Losq, "Referenceless Random Testing", *Digest Int. Fault Tolerant Computing - 6*, Pittsburgh, June 1976, pp. 108-113.

[4] J.P. Hayes, "Transition Counting Testing of Combinational Logic Circuits", *IEEE Trans. on Computers*, Vol. C-25, No. 1, June 1976, pp. 613-620.

[5] M.A. Aboulhamid and E. Cerny, "A Class of Test Generators for Built-in Testing", *IEEE Trans. on Computers*, Vol. C-32, No.10, Oct. 1983,pp.957-959

[6] R. Dandapani, J.H. Patel and J.A. Abraham, "Design of Generators for Built-in Test", *IEEE Int. Test Conf.*, Oct. 1984.

[7] F.C. Hennie, "Fault Detecting Experiments for Sequential Circuits", *Proc. 5th Ann. Symp. Switching Circuit Theory and Logical Design*, Princeton, New Jersey, November 1964.

[8] E.B. Eichelberger and T.W. Williams, "A Logic Design Structure for LSI Testability", *Journal of DA and FT Computing*, Vol. 2, No. 2, May 1978, pp. 165-178.

[9] Fujiwara, et al., "Easily Testable Sequential Machines with Extra Inputs", *IEEE Trans. on Computers*, Vol. C-24, August 1975, pp.821-826.

[10] C.S. Venkatraman and K.K. Saluja, "Transition Count Testing of Sequential Machines", Digest of Papers, *Tenth International Symp. on Fault Tolerant Computing*, Kyoto, Japan, October 1980, pp. 167-172.

[11] D.K. Pradhan, "Sequential Network Design Using Extra Inputs for Fault Detection", *IEEE Trans. on Computers*, Vol. C-32, No. 3, March 1983, pp. 319-323.

[12] S.Z. Hassan, "Signature Testing of Sequential Machines", to appear in *IEEE Trans. on Computers*.

[13] E.F. Moore, "Gedanken-Experiments on Sequential Machines", Automata Studies, Ed. Shanon and McCarthy, Princeton Univ. Press, pp. 129-153.

[14] J. Savir, "Syndrome Testable Design of Combinational Circuits", *IEEE Trans. on Computers*, Vol. C-29, No. 6, June 1980, pp. 442-451 and Vol. C-29, No. 11, pp. 1012-1013.

[15] H. Fujiwara and K. Konoshita, "Testing Logic Circuits with Compressed Date", Digest of Papers, *Eighth International Conference on Fault Tolerant Computing*, Toulouse, France, June 1978, pp. 108-113.

[16] M. Breuer and A.D. Friedman, Diagnosis and Reliable Design of Digital Systems, Computer Science Press, 1976.

[17] K.K. Saluja and R. Danapani, "Testable Design of Sequential Machines Using Checking Experiments", Technical Report EE8416, Department of Electrical and Computer Engineering, University of Newcastle, N.S.W., Australia.

ANALYSIS OF ALTERNATIVE IMPLEMENTATIONS OF PARALLEL SIGNATURE ANALYZERS

Peter C. Maxwell

Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, CA 94720*

ABSTRACT

This paper considers the analysis of parallel signature analyzers from the point of view of finite state switching circuits. This approach is used to investigate and compare the two most common implementations. It is shown that both perform polynomial division, with each input of one circuit being equivalent to a combination of inputs of the other. Differences with regard to error detection capabilities are highlighted. The concept of reconfigurable analyzers is examined and some advantages discussed with regard to regularity of implementation and improved error detection capabilities.

1. Introduction

As integrated circuit packing densities increase, with a corresponding increase in the complexity of circuits which can be fabricated, the use of extra hardware to achieve some degree of built-in testing is becoming more and more widespread. Since the amount of information involved can be very large, data compression techniques are often used to limit the amount of data which has to transferred using the primary inputs and outputs of the system. Apart from the obvious benefit of reduced test time for a given number of test patterns, built-in hardware also enables networks to be exercised at normal systems rates, rather than the much slower rate necessitated by externally loading each individual test pattern. This latter testing may not detect many types of physical failures which give rise to dynamic, position-dependent faults (such as are found in RAM arrays) or failures resulting in timing errors (Banerjee & Abraham, 1983).

In any compression technique, the final data which is inspected may be viewed as the "signature" of the particular node or network under test. Although such a general view may be taken (Bhaskar, 1982), it is more common to use the term "signature analysis" to refer to systems which employ linear feedback shift registers (LFSR's). In such cases the compression is effected by dividing the input data stream, expressed as a polynomial, by the characteristic polynomial of the compressor LFSR (Peterson 1961, Bhavsar & Heckelman 1981). The signature is then the remainder after division, or a linear combination of the constituent remainder bits. Input data to the circuit under test may also be generated by an LFSR, either producing exhaustive patterns (e.g. McCluskey & Bzorgui-Nesbat 1981) or random patterns (Savir et al 1984).

The effectiveness of signature analysis in detecting faults has been investigated for the single input case (Smith 1980) and for the multiple input case where the parallel inputs may be shown to be equivalent to a single input (Sridhar et al 1982, Hassan 1982). This paper

* On leave from the University of New South Wales, Australia

considers other possible configurations of parallel signature analyzers (PSA's), and shows how error detection properties differ.

2. Configurations of Parallel Signature Analyzers

2.1. General Finite State Switching Circuits

A completely general circuit may be described in terms of state matrices, in the same manner as the treatment in Peterson (1961). If we let the register contents (signature) be represented by vector $\mathbf{s}(t)$ at time t, and the data input vector be $\mathbf{d}(t)$, then the state at time $t+1$ can be written as

$$\mathbf{s}(t+1) = \mathbf{s}(t)T + \mathbf{d}(t)D$$

where T is an $r \times r$ matrix representing feedback connections,
D is an $r \times r$ matrix representing input data selection.

Normally D would be the identity matrix, which would serve to connect each input signal to the corresponding register stage.

Furthermore, we can define a circuit's output as

$$q(t) = \mathbf{s}(t)R$$

with R being an n x1 matrix representing bit selection for an output ("result") signal. In the special case where T is such that polynomial division is carried out, $q(t)$ would represent the quotient bit stream. When a circuit is used for signature analysis, $q(t)$ is likely not to be used.

2.2. Representative Example

In order to illustrate points, a specific example will now be considered, viz, Fig. 1a (PSA1) and its dual in Fig. 1b (PSA2). It is stressed that results are in no way restricted to these circuits.

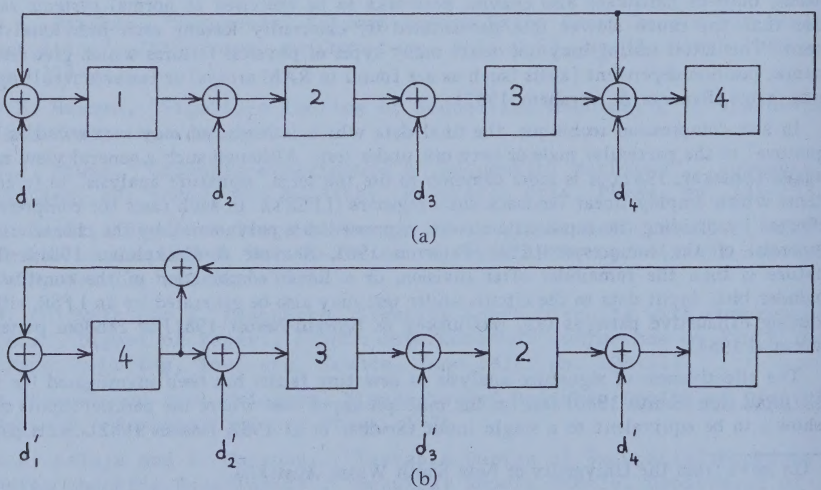


Figure 1. Two configurations of parallel input LFSRs

The characteristic polynomial for both these circuits is

$$f(x) = x^4+x^3+1$$

Vectors and matrices describing PSA2 will be written as primed quantities. For simplicity we will initially take a single input signal, for which case D is a $1xn$ matrix. A single input at the left-most register of PSA1 gives

$$D = [1\ 0\ 0\ 0]$$

and

$$T = \begin{bmatrix} 0\ 1\ 0\ 0 \\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 1 \end{bmatrix}$$

whereas for PSA2, $D' = [0\ 0\ 0\ 1]$ and

$$T' = \begin{bmatrix} 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \\ 0\ 0\ 1\ 1 \end{bmatrix}$$

It is possible to describe the relationships between the two circuits using a matrix A. Then $D' = DA$, $\mathbf{s}' = \mathbf{s}A$ and $T' = A^{-1}TA$. In the above example,

$$A = \begin{bmatrix} 0\ 0\ 0\ 1 \\ 0\ 0\ 1\ 1 \\ 0\ 1\ 1\ 1 \\ 1\ 1\ 1\ 1 \end{bmatrix}$$

The co-ordinate system of PSA2 is the opposite to that of PSA1 so that the state of the left-most register is represented by s_4'.

In addition, it is noted that letting $R' = A^{-1}R$ gives $q'(t) = q(t)$, thereby showing that the circuits produce the same quotient. This is also verified by noting that $s_1' = s_4$.

The configuration in PSA1 is relatively easy to analyse, since an input $d_i(x)$ inserted at position i is equivalent to $x^{i-1}d_i(x)$ inserted at position 1 (Peterson, 1961). Thus the parallel inputs are equivalent to a single input at location 1, formed by summing shifted versions of the $d_i(x)$.

For a PSA of the form in PSA2, the nature of the feedback connections prevents a simple single input equivalent from being able to be formed. It is, however, possible to determine mappings between one system and the other. Consider an input at "position 3" in PSA2. This requires an input selection matrix of $D' = [0\ 1\ 0\ 0]$ which gives

$$D = D'A^{-1} = [0\ 1\ 1\ 0]$$

In other words, the single input $d_3'(x)$ in PSA2 is equivalent to the same input applied at both positions 2 and 3 in PSA1. Consequently, the polynomial operation is equivalent to

$$\frac{x+x^2}{x^4+x^3+1}\, d_3'(x)$$

The final signature is related to the remainder after this division process by expressions which are obtained by using the fact that $\mathbf{s}' = \mathbf{s}A$. Thus,

$$s_1' = s_4$$

$$s_2' = s_3+s_4$$

$$s_3' = s_2+s_3+s_4$$

$$s_4' = s_1+s_2+s_3+s_4$$

Generalizing the above to include 4 parallel inputs, D becomes the identity matrix and $D' = A^{-1}$. This results in the 4 inputs being equivalent to a single input

$$d_1' + (1+x)d_2' + (x+x^2)d_3' + (x^2+x^3)d_4'$$

Although the above procedure is general, the results are specific to a particular circuit configuration and a general result such as applies to PSAs of the type in PSA1 cannot be formulated. Notwithstanding this fact, certain observations can be made regarding respective error detection capabilities, and this is considered in section 3.

2.3. Other Configurations

In the previous analysis, the value of A was chosen to give the appropriate transformation between PSA1 and PSA2. The approach implies no restriction on A other than it be nonsingular. As long as matrices T and T' are similar, then the operation of a circuit represented by T' may be described by equivalent operations of a circuit of the form of PSA1.

In theory, therefore, there are a number of different configurations which will effectively perform division of an input stream by the same polynomial, each corresponding to different values of A. Any of these could serve as a means of obtaining data compression. For example, Fig. 2 shows a circuit which is the transformation of PSA1 using the value

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

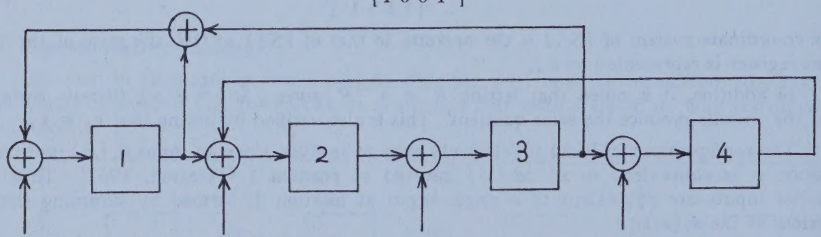


Figure 2. Alternative Transformation of PSA1.

It is noted that the interconnections between the register stages in Fig. 2 are more irregular than those required for either PSA1 or PSA2. For this reason it is unlikely that the alternative configurations would be used in preference to PSA1 or PSA2, unless the different performance with regard to input errors (see next section) was in some way appropriate to expected error types.

3. Application to Error Detection

3.1. Single Bit Errors

The above shows that the equivalent polynomial operation carried out on any input to PSA2 is division by the characteristic polynomial and multiplication by a polynomial whose degree is less than the characteristic polynomial. Consequently, this numerator, when multiplied by an error polynomial of the form $E(x) = x^j$ cannot represent a multiple of the characteristic polynomial, and therefore will result in a different remainder after division to that obtained if $E(x) = 0$. Consequently, single bit errors on any input will be detected in both cases.

3.2. Error Delayed on Different Inputs

By this is meant errors of the form $E_{i+j}(x) = x^{-j} E_i(x)$ i.e. the same error occurring j clock cycles later, j positions further along the register. In PSA1, the quantity added to the correct signature, due to the error terms, is the remainder from the division of the combined error polynomials by the characteristic polynomial. The combined error polynomial is:

$$E(x) = x^{i+j-1}E_{i+j}(x) + x^{i-1}E_i(x) = x^{i+j-1}x^{-j}E_i(x) + x^{i-1}E_i(x)$$
$$= x^{i-1}E_i(x) + x^{i-1}E_i(x) = 0$$

This error cancellation has been previously demonstrated (e.g. Saluja & Dandapani, 1983) and is also easy to visualize by inspecting the circuit. The first incorrect bit (position i) propagates along the shift register and is cancelled at the exclusive-OR gate at position $i + j$ by the arrival of the second incorrect bit. Since the feedback connection is taken only from the end of the register, a variety of error patterns are undetectable, corresponding to different values of i and j.

In circuits of the form of PSA2, the above error cancellation can occur only if positions i and $i + j$ do not have a feedback tap between them. The presence of one or more such taps requires a much more complex, and extremely unlikely, error combination in order to obtain cancellation. For example, in PSA2, an error on d_1' would require a further error on the same input on the next clock cycle in order to cancel the incorrect feedback value, as well as an error on d_2', d_3' or d_4' at the appropriate time to cancel the incorrect data propagating along the register.

This requirement is expressed more formally in the equations derived earlier. Assuming an error x^k occurs on d_1', then the error on d_3' is $x^{-2}x^k$. But the contribution of d_3' has been shown to be $(x + x^2)d_3'$. Thus the error contribution is seen to be

$$E_3(x) = (x + x^2)x^{-2}x^k = (x^{-1} + 1)x^k$$

Thus an additional error of $x^{-1}x^k$ is required on d_1' in order to obtain full cancellation.

3.3. Equally Likely Errors

For PSA1, an error pattern on any particular input is undetectable only if the error polynomial is a multiple of the characteristic polynomial (Smith, 1980). This argument gives rise to the well-known relation that the probability of non-detection of an error, P, is given by

$$P = \frac{2^{k-r} - 1}{2^k - 1} \rightarrow 2^{-r}$$

where r is the number of register stages,
k is the degree of the input polynomial.

For PSA2, we need to consider the equivalent polynomial operation. For example, an error on d_3' would be undetected if $(x + x^2)E_3'(x)$ was a multiple of the charactersitic polynomial, $f(x)$. Depending on the particular $f(x)$, this effect could lead to a reduced error detection rate. The conditions for such an event would be that the term multiplying $E_i'(x)$ was a factor of $f(x)$, resulting in $E_i'(x)$ being required to be a multiple of a polynomial of lower degree than $f(x)$. This effectively reduces the value of r in the above expression for P.

The above effect requires that $f(x)$ be factorable. For configurations where $f(x)$ is irreducible, then the performance of PSA2 is the same as PSA1 for this error type.

4. Reconfigurable Analysers

Section 3.2 demonstrated that cancellation of delayed errors occurs in a circuit of the form of PSA2 only if there are no feedback taps between the respective inputs. Clearly, the way to eliminate the possibility of cancellation is to provide a feedback tap at every position. This would result in characteristic polynomials of the form

$$f(x) = x^r + x^{r-1} + \ldots + x^2 + x + 1$$

for a circuit with r stages.

Such polynomials, although ensuring that one type of error was guaranteed to be detected, would not perform as well as others with regard to other error types, such as cyclic or burst errors. Consequently, improved error detection capabilities would ensue if the feedback taps were made programmable and several signatures collected, each corresponding to a different polynomial. Some advantages in this approach are:

1. Each register stage is identical, since now there is a tap from each output and an exclusive-OR in the feedback path at each stage.
2. In addition to elimination of cancellation of delayed errors as discussed in section 3.2, the probability of non-detection of equally likely errors is improved by rerunning a test and obtaining a different signature. The best reduction occurs if n successive polynomials have no common factors, in which case $P \rightarrow 2^{-nr}$.
3. The same basic circuit can be used to test different modules of a system, each having different characteristics, by choosing polynomials that will maximise the likelihood of detection of a particular error type. This fact, coupled with point 1 above, makes automatic generation particularly straight forward, since the number of stages is the only variable.

In addition to the above, in some cases information regarding a specific error pattern may be obtained. If some signatures indicate a response sequence is correct, which others indicate is incorrect, then it could be stated that the error pattern is a multiple of the polynomials corresponding to the former signatures and is not a multiple of those corresponding to the latter. This information may then be able to be used to aid in the localisation of the error, a process which is very difficult due to the loss of data due to compression. Unfortunately, in general it would require a very large number of signatures before one was obtained where the error pattern was a multiple of the characteristic polynomial.

5. Conclusions

This paper has showed that various configurations of parallel signature analyzers may be considered as linear finite state switching circuits, thereby relating them to the well-known "standard" configuration. Since the error detection properties are different, which type of implementation used may be dependent on a particular application. On the basis of ease of generation and regularity it is most likely that the choice of configuration would be between one which has the feedback formed using exlusive-ORs solely between register stages (PSA1) or solely in the feedback path (PSA2).

A reconfigurable analyzer involves an increased area, since a coefficient register needs to be included with the module. The tradeoff to be made here is with an increased error detection rate.

6. Acknowledgements

This research was supported in part by the Joint Microelectronics Research Centre, University of New South Wales and by DARPA under contract N00039-83-C-0107.

7. References

1. Banerjee, P. & Abraham, J. (1983) "Generating Tests for Physical Failures in MOS Logic Circuits", IEEE Test Conference, pp 554-559.
2. Bhaskar, K. (1982) "Signature Analysis: Yet Another Perspective", IEEE Test Conference, pp 132-134.
3. Bhavsar, D.K. & Heckelman, R.W. (1981) "Self-Testing by Polynomial Division", IEEE Test Conference, pp 208-216.
4. Hassan, S.Z. (1982) "Algebraic Analysis of Parallel Signature Analyzers", CSL TN No 203, Centre for Reliable Computing, Stanford University.
4. McCluskey, E.J. & Bozorgui-Nesbat (1981) "Design for Autonomous Test", IEEE Trans. on Computers, **C-30**, pp 866-875.
6. Peterson, W.W. (1961) *Error Correcting Codes*, MIT Press, Cambridge, Massachusetts.
7. Saluja, K.K. & Dandapani, R. (1983) "Comments on a Data Compression Technique for VLSI Testing", Proc. 21st Allerton Conf. on Communication, Control & Computing, pp 796-805.
8. Savir, J., Ditlow, G.S. & Bardell, P.H. (1984) "Random Pattern Testability", IEEE Trans. on Computers, **C-33**, pp 79-90.
9. Smith, J.E. (1980) "Measures of the Effectiveness of Fault Signature Analysis", IEEE Trans. on Computers, **C-29**, pp 510-514.
10. Sridhar, T., Ho, D.S., Powell, T.J. & Thatte, S.M. (1982) "Analysis and Simulation of Parallel Signature Analyzers", IEEE Test Conference, pp 656-661.

TEST DERIVATION FOR STUCK-OPEN FAULTS IN ITERATIVE LOGIC ARRAYS.

Ravi Sharma Damu Radhakrishnan
Department of Electrical Engineering
Old Dominion University
Norfolk, VA 23508.

ABSTRACT

In this paper we develop testability criteria for stuck-open (s-op) faults in one-dimensional unilateral iterative logic arrays (ILA) of combinational cells. Necessary and sufficient conditions are given for the testability of a s-op fault in a CMOS ILA cell. These conditions are extended to include s-op fault C-testability of the array.

INTRODUCTION

Fault diagnosis in iterative logic arrays (ILA) was studied by many researchers [1-7] in the past because of their direct adaptability to VLSI circuits. They possess the unique characteristics of structural regularity, simplified test derivation and ease of fabrication. One of the problems that confronts the test engineer is the increasing complexity of the test set with the size of the circuit. It has been shown by many researchers that the design of ILA's could be easily modified to enable them to be tested with a constant number of tests independent of the size of the array (C-testability) [2,3,6,7]. These testability measures are based on functional fault models of the cells wherein the circuit may enter into a faulty next state instead of the correct one. This model assumes that a combinational module remains combinational in the presence of a fault. But the use of CMOS and pass logic introduces a new kind of fault called stuck-open (s-op) which produces sequential behavior in an otherwise combinational module [8,9]. Thus testing of CMOS and pass logic cell arrays needs the modification of the above testability criteria to include the effect of these s-op faults. In this paper, we develop testability criteria for s-op faults in one-dimensional unilateral ILA's of

combinational cells under the assumption of a single faulty cell.

FAULT DETECTION IN ILA'S

An ILA consists of a finite number of identical interconnected cells. Each cell receives an input x (referred to as state) from its left-hand neighbor and an external input y. It generates an observable output $\hat{y}$ and transmits the next state $\hat{x}$ to its right-hand neighbor as shown in Fig.1. A cell in the array is characterized by a flow table which has a row for each x input and a column for each y input. Let $X = \{x_1, x_2, .., x_i, .. x_n\}$ be the set of present states and $Y = \{y_1, y_2, .., y_i, .. y_m\}$ be the set of y inputs to the cell flow table. The entries in this table consists of pair of cell outputs $(\hat{x}, \hat{y})$ for each combination (x,y) of the cell inputs. In particular, the entries in row x_i, column y_j will be represented as $(\hat{x}(x_i, y_j), \hat{y}(x_i, y_j))$ and a transition from the present state x_i to the next state x_j under input y_k as $x_i \overset{y_k}{} x_j$. A fault where the erroneous next state reached is x_r instead of the correct next state x_j will be denoted by $x_i \overset{y_k}{} x_j/x_r$. The necessary and sufficient conditions for this fault to be C-testable are given in [2]. In addition, sufficient conditions for the C-testability of an ILA are also given in [2]. This enables the modification of any flow table to make it C-testable. An improved technique using the concept of one-step C-testability is given in [6]. Also given in [6] are the sufficient conditions to make any arbitrary flow table one-step C-testable. Since any flow table that satisfies the conditions of one-step C-testability is also C-testable, the development of C-tests for s-op faults in this paper assumes that the flow table is one-step C-testable.

The detection of s-op faults in combinational circuits and their test generation procedures are given in [10,11]. These procedures employ a pair of test vectors to detect a s-op fault. The first is used to set up the initial conditions required to test the fault. The second is the test which distinguishes between the good circuit and the bad circuit. This two vector approach is used here to derive C-tests for s-op faults in CMOS ILA's.

A basic property used in this research is that all s-op faults in a cell manifest itself onto the output lines of the cell. This is illustrated by the following example. Consider the CMOS NOR gate cell shown in Fig.2(a). S-op faults may occur at different points in the circuit (shown by X's). Consider first the open faults at a,b,c,d&e. Under fault-free conditions if at least one of the inputs x or y is 1, the output will be pulled to 0 and for <x,y> = <00> the output will be pulled to 1. If any of the s-op faults a,b,c,d or e exist, the output will retain the previous logic state for the <00> input. Thus the presence of an open condition at a,b,c,d or e is reflected as a fault on the output - the output cannot be pulled up to logic 1. The presence of an open at j,k or l will not allow the output to be pulled down for <01> input. Similarly, an open at f,g or h will not allow the output to be pulled down for <10> input. Thus the effect of the faults at f,g,h,j,k or l can be stated in general as: the output cannot be pulled down to logic 0. An open at i has the same effect. An open at q or r may not allow the output to be pulled up to logic 1. Moreover, an open at q or r may not allow the output to be pulled down to logic 0 for input <01> or <10> respectively. Hence the effect of this fault can be stated as: neither the output can be pulled up or pulled down. Thus the set of all s-op faults in the cell maps onto a set of faults on the output lines of the cell.

STUCK-OPEN FAULT MODEL

In our approach, the following assumptions are made on the fault model.

1. The basic cell has no $\hat{y}$ output and only the $\hat{x}$ output of the right-most cell is observable.
2. The s-op faults are classified into three types.

a) faults which do not allow the output line of a cell to be pulled up (logic 1) by any input vector applied to the cell. These faults are denoted by z^0.

b) faults which do not allow the output line of a cell to be pulled down (logic 0) by any input vector applied to the cell. These faults are denoted by z^1.

c) faults which do not allow the output line of a cell to be either pulled up or pulled down. These are denoted by z^u.

Using the above fault model, a s-op fault under a transition is represented as $x_i^{y_k}\ x_j/z_p^q$, where $q \in \{0,1,u\}$, $0 \leq p \leq n-1$, and n represents the number of intercell connections (width of $\hat{x}$). Thus $1^3\ 5/z_2^0$ denotes a s-op fault z^0 on line 2 of $\hat{x}$ when the cell makes a transition from state 1 to state 5 under input 3. The $\hat{x}$ lines are numbered from 0 to n-1 and the set of states $X = \{x_1, x_2, .., x_i, .. x_n\}$ are assumed to be binary coded. In this paper the s-op faults considered are restricted to z^0 and z^1.

STUCK-OPEN FAULT TESTABILITY

An array is said to be testable for a s-op fault if it is possible to propagate the effect of this fault to an observable output of the array independent of the location of the fault.

To test for the fault $x_i^{y_k}\ x_j/z_p^q$ in a cell C_q, it is necessary to apply an input pair (x_{in}, y_n) which will drive the pth line of C_q to logic value q. This initializes the cell C_q. Next, with the input combination (x_i, y_k) applied to C_q, the correct next state reached will be x_j. The erroneous next state reached is denoted by $x_r \in X$ which is same as x_j except for the pth bit which will retain the logic value q. A few of the definitions and theorems used in this research are given below.

Definition 1: Two states x_i and x_j of a cell C_q in an array are distinguishable if there exists at least one input vector $Y = y_1 y_2 .. y_m$ that can be applied to the cells following C_q such that the observable outputs of the array are different, depending on whether x_i or x_j is the state of C_q.

Lemma 1: The subset of all s-op faults on the output lines of a cell for all possible state transitions forms a functionally complete set of s-op faults for the cell.

Proof: The proof of this lemma can be illustated by considering the CMOS NOR gate cell shown in Fig.2(a). Its flow table is shown in Fig.2(b). The faults $0^0\ 1/z_0^1$, $0^1\ 0/z_0^0$, $1^0\ 0/z_0^0$ and $1^1\ 0/z_0^0$ are undetectable faults. The effect of the other faults on the cell are as follows. The fault $0^0\ 1/z_0^0$ will not allow the output line $\hat{x}$ to be pulled to logic 1 for the input pair $\langle x,y \rangle = \langle 00 \rangle$. The output can be

pulled to logic 0 if either x or y or both are made 1 and therefore tests for open conditions at a,b,c,d&e. The fault 0^1 $0/z_0{}^1$ will not allow $\hat{x}$ to be pulled to logic 0 for the input pair <01>. A test for this fault in turn tests for the s-op faults at i,j,k&l. Similarly, a test for the fault 1^0 $0/z_0{}^1$ tests for the faults at f,g,h&i. In an analogous manner a test for the fault 1^0 $0/z_0{}^u$ tests for the s-op fault at r and 0^1 $0/z_0{}^u$ tests for the fault at q. Thus the fault set 0^0 $1/z_0{}^0$, 0^1 $0/z_0{}^1$, 1^0 $0/z_0{}^1$, 1^0 $0/z_0{}^u$ and 0^1 $0/z_0{}^u$ test covers all the s-op faults in the cell. The complete set of detectable faults on the output includes two more faults 1^1 $0/z_0{}^1$ and 1^1 $0/z_0{}^u$ and hence the lemma.

Q.E.D.

Lemma 2: A fault $x_i{}^{y_k}$ $x_j/z_p{}^q$ in a cell C_q is testable iff:

a) the cell can be initialized to a state $x_m \in X$ such that its pth bit is q,

b) the pth bit in x_j has logic value q' and

c) x_j and x_r are distinguishable (x_r as defined earlier).

Proof- Necessity: The necessity of condition (a) is obvious from the definition of the fault $z_p{}^q$. Let the logic value on the pth line for x_j be the same as the faulty logic value q. If this is the case, the faulty state and fault-free state happen to be the same and hence contradicts the statement that the fault is testable.

Assume that the fault is testable but x_j and x_r are not distinguishable. This implies that there exists some input sequence $Y = y_1y_2..y_r$ such that $\hat{x}(x_j,Y) = \hat{x}(x_r,Y) = x_s$ (say). It is not possible to determine from x_s whether the pth bit in the y_k-successor of x_i has logic value q or q' and hence a contradiction.

Sufficiency: Since the cell can be initialized to x_m the pth bit in y_k-successor of x_i will be q/q' under faulty/fault-free condition (by a&b). By (c) we can identify whether the pth bit in y_k-successor of x_i was q or q'. Hence the fault $x_i{}^{y_k}$ $x_j/z_p{}^q$ is testable.

Q.E.D.

STUCK-OPEN FAULT C-TESTABILITY

Definition 2: A fault $x_i \overset{y_k}{} x_j/z_p^q$ is C-testable if it is testable and can be tested in every cell of the array by a constant number of tests independent of the length of the array.

A C-test for a s-op fault consists of two test vectors. The first one, the IN-test, applied at time t_1 initializes the cells to the state $x_m \in X$ (as defined in Lemma 2) and the second one, the conventional C-test, applied at time t_2, where $t_2 > t_1$, tests for the fault in the initialized cells.

To initialize the cells $C_1, C_{1+L}, C_{1+2L}, \ldots C_{1+nL}$ to the state $x_m \in X$, there must exist an input state $x_{in} \in X$, an input $y_n \in Y$ and a sequence S of input vectors of length L-1 such that $\hat{x}(x_{in}, y_n) = x_m$ and $\hat{x}(x_{in}, y_n S) = x_{in}$ where $(y_n S)$ is of length L. The sequence $x_{in}, (y_n S)^*$, where $(y_n S)^* = y_n S y_n S \ldots$ is called the IN-test. The above conditions ensure that the input combination (x_{in}, y_n) can be applied to every Lth cell in the array to initialize it to the state x_m.

Now to test for the s-op fault in cells $C_1, C_{1+L}, \ldots, C_{1+nL}$, a C-test is applied which is a sequence of the form $x_i, (y_k R)^*$ where R is an input vector of length L-1 such that $\hat{x}(x_i, y_k R) = x_i$ and $\hat{x}(x_j, (R y_k)^*) \neq \hat{x}(x_r, (R y_k)^*)$. Every cell in the array can be tested by L-1 shifted sequences of these two test vectors.

Theorem 1: A fault $x_i \overset{y_k}{} x_j/z_p^q$ is C-testable if all the following conditions are satisfied:

a) there exists an IN-test to initialize the cells to the state x_m,

b) the fault $x_i \overset{y_k}{} x_j/x_r$ is C-testable, and

c) the length of the IN-test is either equal to or a submultiple of the length of the C-test.

Proof: The IN-test initializes the cells C_1, $C_{1+\ell}$, $C_{1+2\ell}$, $\ldots, C_{1+n\ell}$ to state x_m (pth line to logic value q). Next with the input combination (x_i, y_k) applied to C_1, the correct next state will be x_j. If there is a fault z_p^q the erroneous next state reached will be x_r since the fault will not allow the pth line to go to logic value q', instead, the pth output line will stay in logic state q and the logic values in all other lines will follow correctly. Thus the effect of z_p^q on

the cell is to cause a transition to state x_r instead of to state x_j. A C-test for the transition $x_i^{y_k}\ x_j/x_r$ will test cells $C_1, C_{1+L}, C_{1+2L}, \ldots C_{1+mL}$. To ensure the initialization of these cells, L must be equal to $s\ell$ where s is any integer. For s = 1 both test vectors have the same length. Since the fault $x_i^{y_k}\ x_j/x_r$ is C-testable by (b) it follows that $x_i^{y_k}\ x_j/z_p^q$ is also C-testable.

Q.E.D.

Example 1: For the C-testable flow table shown in Fig.3(a) consider the fault $2^0\ 0/z_1^1$. The cells to be tested for this fault must be initialized to $x_m = 2$ or 3. The IN-test $1,(123)^*$ initializes the cells C_1, C_4 etc. to $x_m = 2$ with the initial state $x_{in} = 1$. Similarly, the IN-test $2,(331)^*$ initializes these cells to $x_m = 3$ with $x_{in} = 2$. To test for this fault the required C-test is $2,(003)^*$. One cycle of the IN-test $1,(123)^*$ and the C-test are shown in Fig.3(b). By two shifted versions of these tests we can test for the above fault in all the cells of the array.

For the fault $1^0\ 0/z_0^1$ the IN-test can be either $0,(00)^*$ which initializes the cells C_1, C_3, C_5, C_7 etc. to $x_m = 1$ with $x_{in} = 0$ or $0,(10)^*$ which initializes these cells to $x_m = 3$ with $x_{in} = 0$. The required C-test is $1,(022200)^*$ which tests for the fault in the cells C_1, C_7 etc. In this case the IN-test length is a submultiple of the length of the C-test.

By Theorem 1, to derive a C-test two test vectors are needed with their lengths related as specified by condition (c). In general, the problem of finding a pair of test vectors in this manner is complex. This can be simplified if we modify condition (b) of Theorem 1 such that $x_m = x_r$.

The following theorem gives sufficient conditions for a s-op fault to be C-testable.

Theorem 2: A fault $x_i^{y_k}\ x_j/z_p^q$ is C-testable if:

a) the flow table is one-step C-testable,

b) a state $x_{in} \neq x_i$ can be found such that for some input y_n, $\hat{x}(x_{in}, y_n) = x_r$, and

c) x_i and x_{in} are distinguishable by y_k.

Proof: A one-step C-testable flow table ensures that 1) the pair graph of the cell flow table is strongly connected, 2) the flow table has no constant entry columns,

3) there is at least one strongly connecting input $y_r \in Y$ and 4) there exists at least one input $y_p \in Y$ and two states $a,b \in X$ such that for some $x_k \in X$ and every $x_s \in S_k$, $\hat{x}(x_k/x_s,y_p) = a/b$ where S_k is the set of all states excluding x_k [6]. The validity of Theorem 2 can be shown by giving a construction procedure for the C-test. The condition $\hat{x}(x_{in},y_n) = x_r$ ensures the possibility of initializing a cell to x_r. When the C-test is applied to C_l, $\hat{x}(x_i,y_k) = x_j$ if there is no fault and $\hat{x}(x_i,y_k) = x_r$ if there is a s-op fault. By the conditions 1,2,3 and 4 of one-step C-testability we can always find inputs such that $\hat{x}(x_j/x_r,y_r^*) = x_k/S_k$, $\hat{x}(x_k/S_k,y_p) = a/b$ and $\hat{x}(a/b,Y_c) = x_i/x_{in}$, where the input vector $Y_c = y_s y_{s+1} \cdots y_{s+t}$ for some t.

If x_i and x_{in} are distinguishable by y_k then $\hat{x}(x_i/x_{in},y_k) = x_j/x_n$, where $x_n \neq x_j$. Now by applying the input y_r repeatedly we can always achieve $\hat{x}(x_j/x_n,y_r^*) = \hat{x}(x_j/x_r,y_r^*) = x_k/S_k$. This produces a C-test $x_i,(y_k y_r^* y_p Y_c)^*$ and an IN-test $x_{in},(y_n y_r^* y_p Y_c)^*$ of the same length. For $y_n = y_k$ both test vectors are the same.

Q.E.D

Example 2: For the fault 0^0 $1/z_0{}^0$, x_r=0. The IN-test is $1,(022200)^*$ and the C-test is $0,(022200)^*$. For the fault 0^0 $1/z_1{}^1$, x_r=3. By selecting x_{in}=2, the IN-test is $2,(2222003)^*$ and the C-test is $0,(0222003)^*$. For the fault 1^0 $0/z_1{}^1$, the only value for x_{in} is x_i since x_r=2. However, if we initialize the cells to x_m=3, we can find an x_{in}=0 to satisfy condition (c) of Theorem 2. The IN-test is therefore $0,(10)^*$ and the C-test is $1,(00)^*$. The fault 2^0 $0/z_1{}^1$ cannot be tested by initializing the cell to x_r=2 since the corresponding x_{in}=1 and x_i=2 are not distinguishable by y_k=0. However if we add a new column to the flow table and select an entry of 2 for the first row, it can be made C-testable.

CONCLUSION

Sufficient conditions for the s-op fault C-testability on the flow table of an ILA cell are presented in this paper. A procedure to derive test vectors for the C-testability of a s-op fault has been developed. It is shown that any flow table can be made s-op fault C-testable by adding extra columns to the cell flow table.

REFERENCES

[1] W.H.Kautz,"Testing for faults in cellular logic arrays," in Proc. 8th Annu. Symp. Switching Automat. Theory, pp. 161-174, 1967.

[2] A.D.Friedman,"Easily testable iterative systems," IEEE Trans. Comput., vol. C-22, pp.1061-1064, Dec. 1973.

[3] F.J.O.Dias,"Truth-table verification of an iterative logic array," IEEE Trans. Comput., vol. C-25, pp. 605-613, June 1976.

[4] B.A.Prasad and F.G.Gray,"Multiple fault detection in arrays of combinational cells," IEEE Trans. Comput., vol. C-24, pp. 794-802, Aug. 1975.

[5] F.G.Gray and R.A.Thompson,"Fault detection in bilateral arrays of combinational cells," IEEE Trans. Comput., vol. C-27, pp. 1206-1213, 1978.

[6] R.Parthasarthy and S.M.Reddy,"A testable design of iterative logic arrays," IEEE Trans. Comput., vol. C-30, pp. 833-841, Nov. 1981.

[7] T.Sridhar and J.P.Hayes,"Design of easily testable bit-sliced systems," IEEE Trans. Comput.,vol. C-30, pp. 842-854, Nov. 1981.

[8] R.L.Wadsack,"Fault modeling and logic simulation of CMOS and MOS integrated circuits," The Bell Systems Tech. Jour., vol.57, No.5, pp. 1449-1474, June 1978.

[9] D.Radhakrishnan and G.K.Maki,"Test derivation for MOS switch logic networks," Proc. of the 21st Annual Allerton Conf. on Communication Control and Computing, Monticello, ILL, pp. 786-795, Oct. 1983.

[10] R.Chandramouli,"On testing stuck-open faults," Proc. of Fault Tolerant Computing Symp., IEEE, pp.258-265, 1983.

[11] Y.M.El-Ziq,"Automatic test generation for stuck-open faults in CMOS VLSI," 18th Design Aut. Conf., pp. 347-352, June 1981.

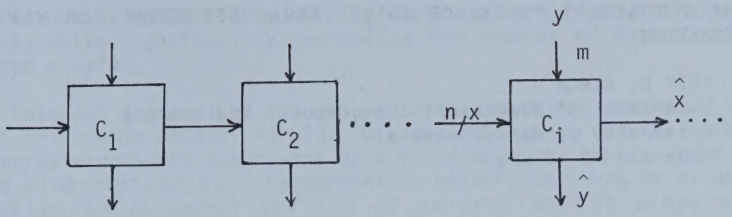


Fig. 1. ONE-DIMENSIONAL ILA

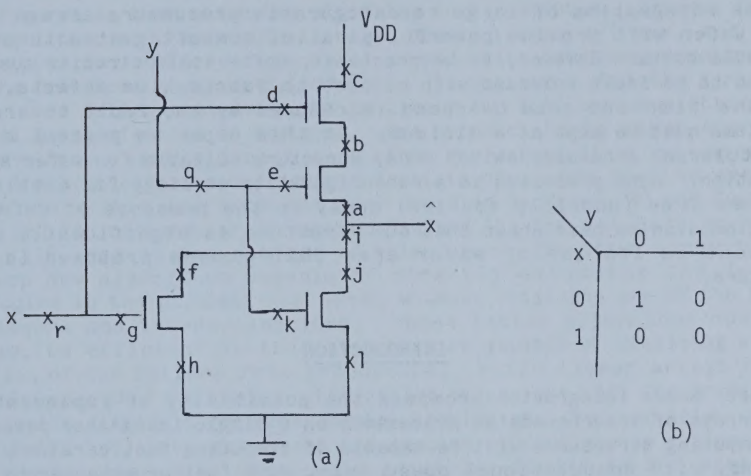


Fig. 2. A CMOS NOR GATE CELL AND ITS FLOW TABLE

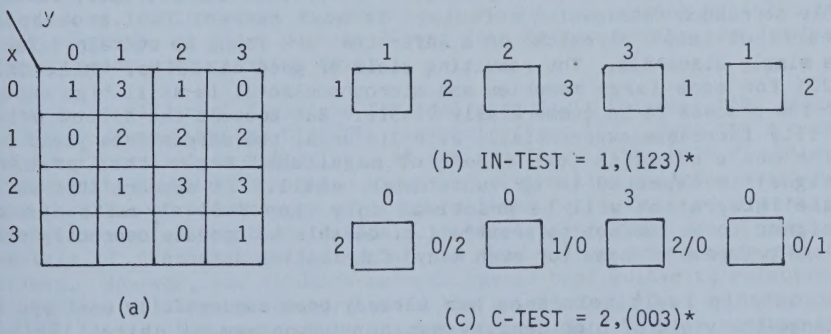


Fig. 3. A ONE-STEP C-TESTABLE FLOW TABLE AND THE TEST VECTORS FOR THE S-OP FAULT $2^0 0/Z_1^1$

A FAULT TOLERANT PROCESSOR SWITCH ARRAY STRUCTURE FOR WAFER SCALE INTEGRATION

ADIT D. SINGH
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

ABSTRACT

The integration of large reconfigurable processors arrays on a single wafer will provide powerful parallel computing structures at reasonable costs. However, to be practical, wafer scale circuits must be designed to be fault tolerant with respect to fabrication defects. At the same time the area overhead introduced by the fault tolerance mechanisms must be kept at a minimum. In this paper we present a new fault tolerant processor switch array structure suitable for wafer scale integration. Also presented is a reconfiguration strategy for obtaining a failure free (possibly smaller) array in the presence of defects. Simulation studies have shown that our structure is significantly more efficient in its use of wafer area than others proposed in the literature.

I. INTRODUCTION

Wafer Scale Integration promises the possibility of implementing large arrays of interconnected processors on a single integrated package. Such computing structures will be capable of executing fast parallel algorithms, with computational speed being even further enhanced by the fact that interprocessor signals will not need to be driven off chip. Unfortunately, however, large area VLSI circuits are extremely susceptible to random fabrication defects. In most current VLSI processes, dies (individual circuits) on a wafer that are found to contain defects are simply discarded. The resulting yield of good circuits, while only 5-20% for some large memories and microprocessors, is still high enough for the process to be commercially viable. But because the defect probability increases exponentially with die area, the defect free yield for wafer scale circuits (two orders of magnitude larger than present designs) is expected to be vanishingly small. It appears that wafer scale integration will be practical only when VLSI circuits can be designed to be "defect tolerant", i.e. be able to operate correctly even in the presence of some (or even many) fabrication defects.

On chip fault tolerance has already been successfully employed to enhance the yield of high density semiconductor memory chips [1]-[3]. These are implemented with spare word and bit lines which are used, as needed, to replace others disabled by fabrication defects. The personalization is done by using a laser to blow fuses or with electrically programmable links. This approach is effective because, due to the regular structure of memory cell arrays, a single spare bit or word line can act as a non dedicated spare for a large part of the circuit. The

amount or redundancy added to the chip is small, which increases the area only slightly while significantly increasing the number of "good" dies obtained from a wafer.

Fault tolerant designs are also being investigated for implementing large VLSI processor arrays [4]-[9]. Of particular interest are reconfigurable array structures comprising of a two dimensional host grid of processing elements and an interconnection network that can be dynamically configured to embed any one of several useful processor interconnect topologies. An example is the multiprocessor array presented by Koren [4]. For a given computation, the host array is reconfigured to best match the algorithm to be executed. Schemes for embedding linear and rectangular arrays, binary trees, x-trees, and other topologies for efficient parallel computation have been developed for failure free host arrays [4],[10]-[12]. In the event that the array contains some failed processors, as is likely for VLSI implementations, two general strategies can be employed. One approach is to first extract a failure free (virtual) host array from the actual (physical) array, in which the application based computational topology can then be embedded as before. This approach has been studied by Koren [4] and Gupta and Zorat [6] for rectangular arrays and Hedlund [5] for the CHIP architecture [13], and is the subject of this paper. The other approach is to develop new algorithms capable of directly embedding the desired topologies in the physical host array without utilizing any of the failed processors and interconnections. These latter algorithms must, of course, be efficient in that they must be capable of utilizing a large fraction of the failure free processors. While linear arrays can be readily embedded in arrays with failed processors [4], the problem appears quite difficult for more complex topologies. For example, Varman and Fussell [15] have shown that the problem of configuring a binary tree in a rectangular host array with arbitrary faults is N-P complete. Another difficulty with this second approach is that the interprocessor distances can become arbitrarily long. This can very adversely affect performance, particularly in synchronous structures such as systolic arrays [14] where the clock must be slowed down to accomodate the longest interprocessor communication delay.

A quite different technique for realizing fault tolerant VLSI arrays is being pursued by the DIOGENES research effort [7]. Here the processors are fabricated in a (logical, if not physical) line, with "bundles" of uncommitted interconnection wires and controlling switches available along the line of processors. Processors found fault free upon testing are then connected together to obtain the desired topology. An important advantage of this approach is that all fault free processors can be utilized. However, the DIOGENES methodology is best suited to relatively small arrays of large processors where the interconnect and switch complexity is small as compared to the processor size. This is because the designs are only fault tolerant with respect to processor failures; interconnect or switch failures can be catastrophic. The probability of these latter failures must, therefore, be kept very small. Also, interprocessor connections for commonly employed topologies such as binary trees and square grids can get quite long for large arrays, resulting in degraded performance.

In this paper we present a new fault tolerant processor switch array structure that can be easily configured (by appropriately setting the switches) to a rectangular failure free host grid in which desired application dependent topologies can then be embedded. Some attractive features of the proposed processor switch structure include:

- o the ability to incorporate a very high proportion of the operational processors in the physical array into the failure free virtual host array
- o the ability to handle both processor and switch/interconnection failures
- o a small fixed bound on the interprocessor connection lengths for arbitrarily large arrays

In the next section we review the work of Koren [4], Hedlund [5] and Gupta and Zorat [6] that addresses the same general problem. We then introduce the proposed processor switch fault tolerant array structure in Section III, and present a procedure for obtaining a failure free virtual host array from a physical array containing faulty processors. Section IV considers the possibility of switch and interconnection failures. The paper concludes in Section V with a discussion of heuristics to speed up the reconfiguration algorithm.

II. RELATED WORK

In [4] Koren proposes obtaining a failure free virtual square grid of processors by eliminating the rows and columns in the physical array that contain failed processors as shown in Figure 1. The failure free processors in the disabled rows and columns act as connecting elements to maintain the required square grid topology. If it is possible to wire around a processor as shown in Figure 2, (note that this implies failure free or redundant interconnections) then only a row or a column need be eliminated for a failed processor. The simplicity of this row/column exclusion approach is clearly very attractive, but the overhead in terms of unutilized good processors is high. For this strategy to be practical the array must contain very few faulty processors. Unfortunately, this cannot be expected of wafer scale circuits as each is statistically likely to have many fabrication defects.

Hedlund [5] proposes a solution to this problem with a two level strategy that involves introducing sufficient redundancy in each processor, or block of processors, to greatly reduce their probability of failure. Specifically, for the CHIP architecture [13] the proposal suggests constructing the host array out of blocks of 2x2 subarrays of processors. These blocks can be themselves made fault tolerant through massive redundancy to ensure to a high probability that each subarray is functional. Such a host array will then contain very few failed blocks, which can be purged through row/column exclusion. While practical in principle, the approach is very inefficient in its use of processors because of the massive redundancy needed to protect the subarrays. In his

examples Hedlund proposes fabricating 12 processors in each block to ensure a failure free 2x2 subarray. On the average, this results in less than half the good processors being actually utilized in the failure free array. Furthermore, there is substantial additional overhead in switches and interconnections.

Gupta and Zorat [6] take another approach in their attempt to make the row/column exclusion strategy more efficient. Their approach involves providing additional connections among the processors in the array, as shown in Figure 3 (for a fan out of 3). Now, in some cases, only a single row or column need be eliminated for multiple processor failures even when they do not lie in the same row or column. The simulation results in [6], however, indicate that while significantly better than simple row/column exclusion, this approach is only marginally more efficient in its use of processors than Hedlund's two level strategy.

Preliminary simulation studies have shown that the new processor switch structure presented here can utilize more than 80% of its fault free processors for large arrays. It does not require significantly greater interconnect complexity to support the reconfiguration. In addition it can tolerate switch and interconnection failures, and retain short interprocessor connection lengths despite such failures.

III. THE PROPOSED PROCESSOR SWITCH ARRAY STRUCTURE

The proposed structure consists of interwoven arrays of switches and processors as shown in Figure 4. Each processor can be connected to any of its four neighboring switches, while the switches are interconnected in the desired rectangular mesh topology. A grid of mesh connected processors can be obtained by associating a unique processor with each of the switch nodes. Note that the structure has redundant processors since the processor array contains an extra row and column. For an nxn switch array there are (n+1) x (n+1) processors, a redundancy of 2n+1.

To understand how a failure free mesh connected virtual grid of processors can be extracted from this physical structure, let us first assume that the switches and interswitch connections are failure free. (We shall later relax this assumption). Then the desired failure free virtual array can be obtained by finding a one to one and onto mapping from the failure free processors to the switches. For an nxn switch array, if the number of failed processors is greater than the number of redundant processors, 2n+1, such a mapping clearly does not exist; even for the converse case it may not always exist. In such cases additional processor redundancy is generated by reducing the size of the switch array. This is done by eliminating a row (or column) and setting all switches in that row to act as connecting elements, as shown in Figure 5. The number of redundant processors is increased by exactly the number of switches in the disabled row, because processors no longer have to be associated with these switches. By alternately eliminating rows and columns (to maintain a square grid) the array can be reduced sufficiently in size so as to make the number of active switches less than (or equal

to) the number of failure free processors. In some cases, because of the distribution of failed processors, additional rows and/or columns may need to be eliminated before all switches can be matched with a unique failure free processor.

Consider an array with nxn switches and (n+1) x (n+1) processors. If f processors in the array are faulty, then at least d = f - (2n+1) switches must be removed before a unique processor can be assigned to each switch. Suppose we achieve this by eliminating an equal number of rows and columns in the switch array to retain a square array. Let this number be k, then k must be greater than or equal to the unique integer k_{min} that satisfies the following expression:

$$n^2 - (n-k_{min})^2 \geq d > n^2-(n-k_{min}+1)^2$$

A match between the switches and processors can be found by representing the array as a bipartite graph as illustrated in Figure 6, and then employing well known matching algorithms [16]. The switches make up one set of nodes in the bipartite graph, the processors the other. An edge connects a node representing a processor to that representing a switch if and only if the processor can be associated with that switch. A match that covers all the switches is the desired mapping.

Since at least k_{min} rows and columns (each) of the switch array must be eliminated to free enough processors to allow a complete match, we begin by trying, in turn, each configuration with k_{min} rows and columns eliminated. If none of these combinations provides a complete match we repeat the process with an additional row and column eliminated, until a match covers all the switches. The resulting mapping defines the largest fault free virtual array that can be extracted from the physical structure.

Because of the flexibility of association among the processors and switches allowed by our proposed array structure, simulation results show that in many cases eliminating k_{min} rows and columns yields a complete match between the active switches and failure free processors. Eliminating k_{min} + 1 rows and columns is sufficient in almost all cases. Additional rows and columns need to be eliminated only for very unique combinations of processor failures that are extremely unlikely to occur in the expected random failure environment.

Note that in the large majority of cases our procedure is expected to yield a failure free virtual square array at most one row and column smaller than the largest possible from the failure free processors. In large arrays, a single row and column contain only a small fraction of the array nodes; for arrays larger than 20x20, this fraction is less than 10%. Thus, the utilization of failure free processors in our approach is high.

Switch/interconnection failures

So far we have made the assumption that the switches and interswitch connections do not fail. In practice, however, these too may experience failures, although being very much simpler than the processors, their failure probabilities can be reasonably expected to be significantly lower. Interconnection failures are handled by eliminating the row or column containing the failing link from the switch array; switch failures are purged by eliminating both the row and the column containing the switch from the array. In both cases the array size is reduced, and the number of redundant processors available is increased. If, as expected, the probability of a switch or interconnection failure is 2% to 10% of that for a processor, then for every switch and interconnection failure one can expect several processor failures. These would anyway require elimination of some rows and columns in the switch array before all the switches can be matched by good processors. The switch or interconnection failures only pin down which rows and columns must be eliminated. Because of the lost flexibility of freely picking these rows and columns to best achieve a match, we do expect some inefficiency. That is, at times it may be only possible to extract a smaller failure free virtual array for a set of processor failures than what might be possible if switch and/or interconnection failures do not predetermine some of the rows and columns to be eliminated. However, preliminary simulations show that this overhead to be small.

V. Discussion

In describing the search procedure for finding a match, we stated that all configurations of the array with k_{min} rows and columns eliminated be tested for a match. We can speed up the search by first searching those configurations that are more likely to yield a match. This can be done, for example, by recognizing that a good candidate for elimination is a switch row (column) for which the two adjacent processor rows (columns) together have the largest number of processor failures. Also, it would very rarely be worthwhile to eliminate two adjacent rows (columns) of switches because all processors in the row (column) between the two would be isolated in the array. The only time this can be done with loss of failure free processors is when all processors in a processor row or column have failed; clearly an extremely unlikely event for large arrays with useful processor survival probabilities. The fact that adjacent switch rows or columns are very unlikely to be eliminated by our match search procedure implies that for almost all resulting failure free virtual arrays, the longest interconnection between adjacent processors will include only two interswitch distances. This is a significant improvement from a performance standpoint over the other proposals for fault tolerant array structures [6,7] that can have arbitrarily long interprocessor connections.

REFERENCES

1. E. Tammaru and J.B. Angell, "Redundancy for LSI yield enhancement," IEEE J. Solid-State Circuits, vol. SC-2, pp. 172-182, Dec. 1967.

2. S.E. Schuster, "Multiple word/bit line redundancy for semiconductor memories," IEEE J. Solid State Circuits, vol. SC-13, pp 698-703, Oct. 1978.

3. T. Mano, M. Wada, N. Ieda and M. Tanimoto, "A Redundancy Circuit for a Fault-Tolerant 256-K MOS RAM," IEEE Jour. of Solid-State Circuits, SC-17, 4 (Aug. 1982), 726-730.

4. I. Koren, "A reconfigurable and fault-tolerant VLSI multiprocesor array," in Proc. 8th Int. Symp. Comput. Architecture, 1981.

5. K.S. Hedlund, "Wafer Scale Integration of Parallel Processors," Technical Report CSD-TR-422, Purdue University, 1982.

6. V. Gupta and A. Zorat, "On Constructing a Grid of Processors for Wafer Scale Integration," TR-83/061, Department of Computer Science, SUNY, Stony Brook, New York.

7. A.L. Rosenberg, "The Diogenes Approach to Testable Fault-Tolerant Arrays of Processors," IEEE Transactions on Computers, vol. C-32, No. 10, October 1983, pp. 902-910.

8. F.T. Leighton and C.E. Leiserson, "Wafer-scale integration of systolic arrays," in Proc. 23rd IEEE Symp. Foundations of Comput. Sci., 1982, 297-311.

9. J.W. Greene and A. El Gamal, "Area and delay penalties in restructurable wafer-scale arrays," in Proc 3rd Caltech Conf. on VLSI, 1983.

10. D. Fussell and P. Varman, "Fault-tolerant wafer-scale architectures for VLSI," in Proc. 9th Int. Symp. Comput. Architecture, 1982.

11. D. Gordon, I. Koren, and G.M. Silberman, "Embedding tree structures in fault-tolerant VLSI hexagonal arrays," IEEE Transactions on Computers, January 1984.

12. R.P. Brent and H.T. Kung, "On the Area of Binary Tree Layouts," Informations Processing Letters, Vol. 11, No. 1, August 1980.

13. L. Snyder, "Introduction to the Configurable, Highly Parallel Computer," Computer, vol. 15, No. 1, January 1982, pp. 47-56.

14. H.T. Kung, "Why Systolic Architectures," Computer, vol. 15, No. 1, January 1982, pp. 37-46.

15. P.J. Varman and D.S. Fussell, "Realizing Fault Tolerant Binary Trees in VLSI," 20th Allerton Conference, 1982.

16. C.H. Papadimitriou and K. Steiglitz, "Combinational Optimization: Algorithms and Complexity," Prentice Hall, NJ 1982.

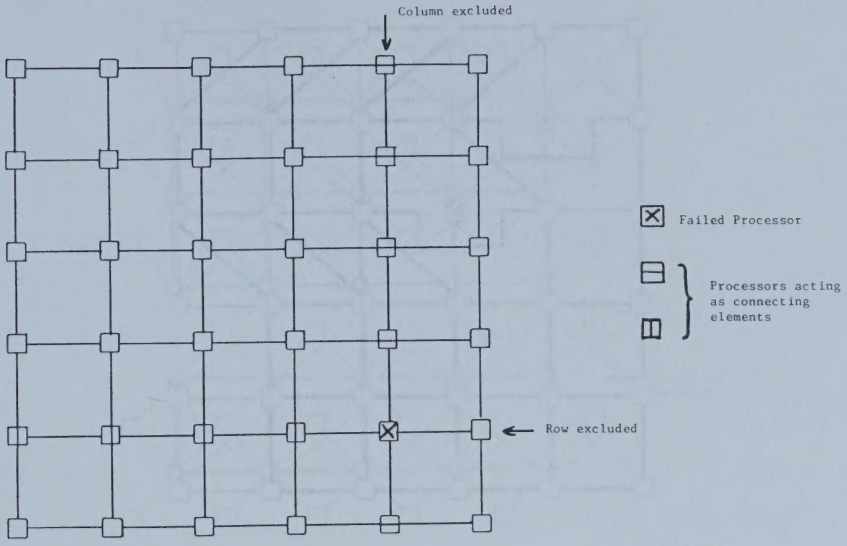


Figure 1: Row and Column Exclusion Strategy

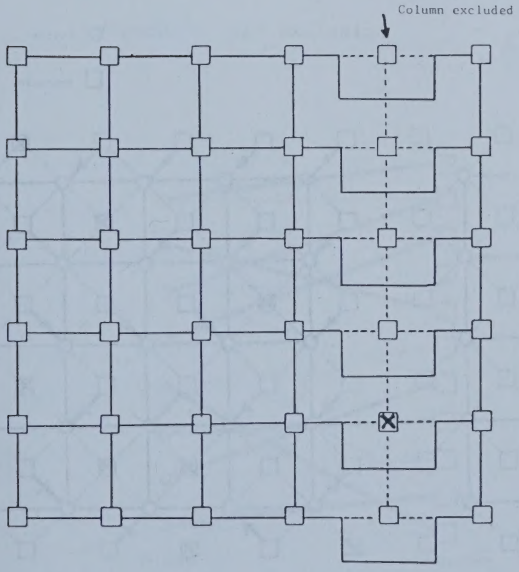


Figure 2: Row or Column Exclusion Through Wire Around Capability

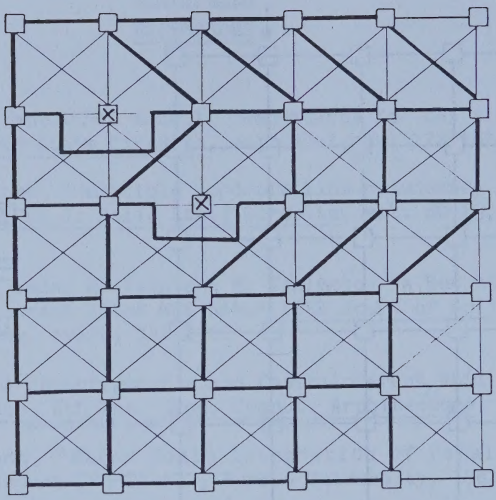
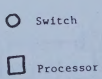


Figure 3: Gupta and Zorat's Array With Fan-out of 3.



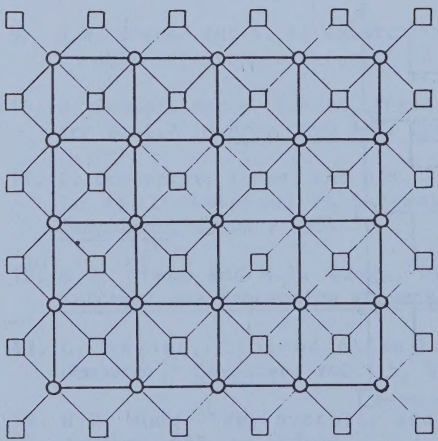


Figure 4(a): The Proposed Array Structure

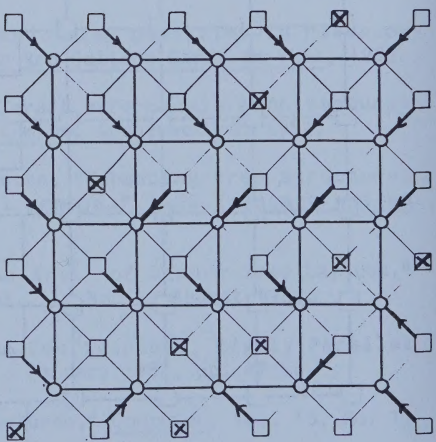


Figure 4(b): Associating Processors With the Switches

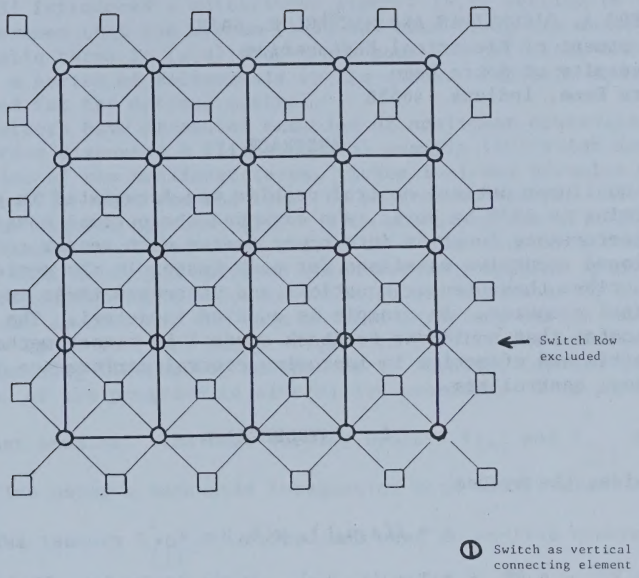


Figure 5: Row Exclusion

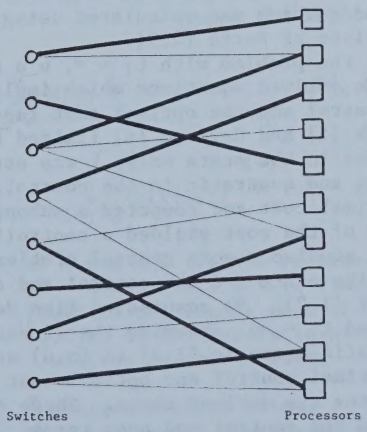


Figure 6: Bipartite Graph Representing the Processor Switch Array, and Showing a Matching

NONLINEAR FEEDBACK DESIGN: OPTIMAL RESPONSES BY TENSOR ANALYSIS*

JOSEPH A. O'SULLIVAN AND MICHAEL K. SAIN
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

The nonlinear optimal control problem was formulated in a tensor algebra setting in 1978 by Buric, who expanded the optimal control and the optimal performance function into power series with tensor coefficients and developed recursive equations for each tensor in the series. This paper describes these tensor equations and discusses their implementation on a digital computer. An example is studied in detail. The investigation indicates that nonlinear feedback control by tensor methods can be both feasible and effective in improving control performance over standard local linear controllers.

I. INTRODUCTION

Consider the system

$$\dot{x} = f(x,u,t), \; x(t_o) = x_o, \tag{1.1}$$

where $x \in R^n$, $u \in R^m$, $t \in T = [t_o,t_1]$. It is desired to find the control u as a series of polynomials in the states so as to minimize the scalar cost

$$J = M(x(t_1)) + \int_{t_o}^{t_1} L(x,u,t)dt. \tag{1.2}$$

This paper discusses the solution to this problem in the tensor algebra setting. An example is included which was calculated using software [1] developed using equations derived by Buric [2,3].

Al'breckht [4] looked at the problem with $t_1 = \infty$, u a scalar and no time dependence of f or L. He derived equations which implicitly yielded expansions for the optimal control and the optimal cost function. Garrard, McClamroch, and Clark [5] and Garrard [6] limited their studies to problems with nonlinearities in the state only; L was assumed to be positive definite in the state and quadratic in the control. They assumed a special form for the suboptimal cost and computed a suboptimal controller. The choice of this form of the cost yielded a controller which was easy to calculate. Lukes [7] studied a more general problem. He assumed there were nonlinearities in the state and the control and cross terms in both the system (1.1) and cost (1.2). He assumed no time dependence in (1.1) or (1.2) and also assumed $t_1 = \infty$. Assuming the linear part of (1.1) is stabilizable and the quadratic terms in (1.2) in (x,u) are positive definite, expansions for the optimal control and optimal cost in terms of homogeneous powers of the states are derived there. These expansions, however, only implicitly yield the control and cost terms. Several equations must be solved simultaneously, making calculations tedious.

*This work was supported in part by the National Aeronautics and Space Administration under Grant NSG 3048.

Buric [2,3] introduced a multilinear algebra [8,9] setting to the problem. He assumes that the system (1.1) and cost (1.2) are analytic, that the quadratic terms in (x,u) from L are positive definite, and that the control is a series of polynomials in the state. Explicit equations are then derived for the optimal control.

Several authors have presented examples of nonlinear controllers. Garrard and Jordan discussed a flight control example [10] which demonstrated the value of the nonlinear terms. Lukes included examples with his derivations [7]. Hill was the first to present an example calculated using the tensor algebra approach [11]. With the exception of Hill's example, every example mentioned above relied upon time consuming hand calculations. Hill developed software which was very problem specific. O'Sullivan [1] then developed software which solved equations (2.8) and (2.9) in the next section for the case of time invariant system and cost tensors for m = 2,3,4 in (2.8) and j = 1,2,3 in (2.9). The software was developed emphasizing flexibility so that it is straight forward to switch the application of the programs to alternative problems. This paper reports these last studies. The time varying tensors $V_{[m]}$ and $K^{1}_{[j]}$ are calculated off-line using a backwards integration with final values for $V_{[m]}$ being $M_{[m]}$. The tensors $K^{1}_{[j]}$ are stored and used in on-line control. An example calculated using this software is included.

II. SOLUTION

We now briefly summarize the formalism introduced by Buric. Assuming that the system (1.1) and cost (1.2) are analytic, these equations may be rewritten as

$$\dot{x} = Ax + Bu + \sum_{p+q>2} A^{1}_{[p],[q]} \overset{p+q}{\circledcirc} u^p \otimes x^q \tag{2.1}$$

$$J = \sum_{j=2} M_{[j]} \overset{j}{\circledcirc} x(t_1)^j + \int_{t_o}^{t_1} \{x^TQx + u^TRu + x^TCu + \sum_{i+j>2} Q_{[i],[j]} \overset{i+j}{\circledcirc} u^i \otimes x^j\}dt, \tag{2.2}$$

where

$u^p = u \otimes u \otimes \ldots \otimes u$ (p times),
$x^q = x \otimes x \otimes \ldots \otimes x$ (q times),
(A,B) is the linear part of (1.1),
(Q,C,R) is the quadratic part of (1.2), written in matrix form,

^{T} denotes transposition,
$\otimes$ is the symbol for tensor products [1,2,8,9],

brackets around subscripts denote symmetry with respect to those indices [1,2,9],

$\circledcirc$ is the contraction operator used to sum over the indices of the associated tensors [1,2],

and

all tensors and matrices (except $M_{[j]}$) may depend on time.

Note that $A^{1}_{[p],[q]} \in R^n \otimes (\otimes^p R^{m*}) \otimes (\otimes^q R^{n*})$ (where * denotes the dual space and $\otimes^p$ denotes the pth tensor product of the associated space) maps elements of $(\otimes^p R^m) \otimes (\otimes^q R^n)$ to R^n via the contraction operation, and $Q_{[i],[j]}$ maps elements of $(\otimes^i R^m) \otimes (\otimes^j R^n)$ to R via $\odot^{i+j}$. Assume that the control is fed back in the form of a series of polynomials in the state, as in [4] and [7], but write it in tensor notation [1-3]

$$u(t) = k(x(t),t) = \sum_{j=1} K^{1}_{[j]} \overset{j}{\odot} x^j. \tag{2.3}$$

Define the performance function $V_k(x(t),t)$ by

$$V_k(x(t),t) = M(x(t_1)) + \int_t^{t_1} L(x,k(x,\tau),\tau)d\tau, \tag{2.4}$$

where x follows from (1.1) and (2.3). Then we have the Hamilton-Jacobi-Bellman equation

$$\frac{dV(x(t),t)}{dt} + L(x(t),k(x(t),t),t) = 0 \tag{2.5}$$

for the optimal cost V(x(t),t), provided that k is chosen to minimize

$$\frac{dV}{dt} + L. \tag{2.6}$$

Substituting Equations (2.1), (2.2), and (2.3) into Equations (2.5) and (2.6) yields equations for V(x(t),t) and the tensors $K^{1}_{[j]}$. Furthermore, in Equation (2.5), no tensor powers of x lower than 2 appear and the following lemma is established [1,2].

Lemma: The optimal performance function is a polynomial in x starting with second order terms

$$V(x(t),t) = \sum_{i=2} V_{[i]} \overset{i}{\odot} x^i \tag{2.7}$$

(here $V_{[i]}$ depends on time).

The following equations give explicit solutions for the tensors $V_{[i]}$ and $K^{1}_{[j]}$:

$$\dot{V}_{[m]} + \sum_{k=2}^{m} \sum_{q=0}^{m+1-k} \sum_{p=\max(1-q,\min(n,1))}^{n} (k\, V_{[k]} \overset{1}{\odot} [(p,q)\, A^{1}_{[p],[q]} \overset{p}{\odot} K^{p}_{[n]}])$$

$$+ \sum_{j=0}^{m} \sum_{i=\max(2-j,\min(m-j,1))}^{m-j} [(i,j)Q_{[i],[j]} \overset{i}{\odot} K^{i}_{[m-j]}], \qquad (2.8)$$

where $n = m-k+1-q$,
(p,q) is a permutation operator which allows the contraction over the p indices to be performed before that over the q indices,

and $K^{p}_{[n]} = \sum K^{1}_{[i_1]} \otimes K^{1}_{[i_2]} \otimes \ldots \otimes K^{1}_{[i_p]}$
where the sum is over all $i_1+i_2+\ldots+i_p=n$;

$$K^{1}_{[j]} = -\frac{1}{2} R^{-1} \overset{1}{\odot} \{(j+1)\, B^{T} \overset{1}{\odot} V^{1}_{[j]} + \sum_{k=2}^{j} \sum_{n=0}^{j-k+1} \sum_{p=\min(2,n+1)}^{n+1} kp[(p,q)$$

$$A^{T}_{[p],[q]} \overset{p}{\odot} (V^{1}_{[k-1]} \otimes K^{p-1}_{[n]})] + \sum_{i=3}^{j+1} i\, Q^{1}_{[i-1],0} \overset{i-1}{\odot} K^{i-1}_{[j]} +$$

$$\sum_{m=1}^{j} \sum_{i=\min(2,j-m+1)}^{j-m+1} [i(i-1,m)\, Q^{1}_{[i-1],[m]} \overset{i-1}{\odot} K^{i-1}_{[j-m]}]\} \qquad (2.9)$$

where $q = j-k-n+1$

and if $x = (x_1,x_2,\ldots,x_n)^T \in R^n$ and $x^* = (x_1,x_2,\ldots,x_n) \in R^{n*}$

then $V^{1}_{[j]}$ is found to satisfy

$$x^* \overset{1}{\odot} V^{1}_{[j]} \overset{j}{\odot} x^j = V_{[j+1]} \overset{j+1}{\odot} x^{j+1}$$

$Q^{1}_{[i-1],[m]}$ is found to satisfy

$$u^* \overset{1}{\odot} Q^{1}_{[i-1],[m]} \overset{i+m-1}{\odot} u^{i-1} \otimes x^m = Q_{[i],[m]} \overset{i+m}{\odot} u^i \otimes x^m$$

and $(p,q)\, A^{T}_{[p],[q]}$ is found to satisfy

$$u^* \overset{1}{\odot} (p,q)\, A^{T}_{[p],[q]} \overset{p+q}{\odot} x^q \otimes y \otimes u^{p-1} =$$

$$y^* \overset{1}{\odot} A^{1}_{[p],[q]} \overset{p+q}{\odot} u^p \otimes x^q,$$

for all $y \in R^n$, $x \in R^n$, $u \in R^m$.

III. EXAMPLE

The system studied is an example which has been used by others for identification purposes [12-15]. This system was devised originally to illustrate a case having nontrivial nonlinearities whose linear approximation was stable and whose equilibrium point was the origin. The problem has been found to have finite escape time which rules out use of geometric control theory according to Brockett [16, p. 63]. Thus the demonstration of the increase in performance brought about by computing nonlinear terms in this example shows the wide range of applicability of the method and the benefits of the increased complexity of the controller.

For the system

$$\dot{x}_1 = u_2 \cosh x_1 x_2 - e^{2u_1} \sinh 2x_1 - 3 \sinh x_2$$

$$\dot{x}_2 = e^{u_1 u_2} \sinh x_1 - e^{u_1} u_1 \cosh x_1^2 + \sinh x_2$$

$$x(0) = x_0 \tag{3.1}$$

minimize the cost functional

$$J = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 2 \end{bmatrix} \circledast x\,(5)^2 + \int_0^5 [x^T \quad u^T] \begin{bmatrix} 2 & 0 & 3 & 0 \\ 0 & 2 & 0 & 3 \\ 3 & 0 & 5 & 0 \\ 0 & 3 & 0 & 5 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} dt. \tag{3.2}$$

Selected feedback gains are plotted over time in Figure (3.1). Example state and control trajectories are included in Figures (3.2) and (3.3). For this problem, a grid of initial conditions in the phase plane at .1 intervals was set up. For each initial condition, the system was integrated out to 5 sec. From the information calculated, "regions of usefulness" were established for the system for three types of feedback: linear only, linear and quadratic, and linear, quadratic and third order. Points were said to be in the region of usefulness if the norm of the state remained finite throughout the interval and if the final norm of the state was less than 0.1. These regions are shown in Figure (3.5). The validity of this condition for a point being inside these regions was established by examining state and control trajectories for a wide variety of initial conditions (see for instance Figure 3.3). Both the third order and second order feedback regions are much larger than the linear feedback region. The third order region is somewhat larger than the second order region and the third order trajectories are dramatically better than the second order trajectories (go toward zero faster, lower norms over most intervals, see plots). Thus the benefit of the nonlinear terms for this example is established.

In applications it is anticipated that the control system designer will not have access to the actual system tensors (as is assumed above). He will be given a model of the system obtained by some approximation or identification scheme. It is this model which will be used to calculate the controls which will be used on the actual system. With this in mind, a model obtained by Yurkovich [13] and Bugajski [14] was used to calculate the feedback tensors for this example. The model is in Figure (3.4). The feedback tensors were used to control system (3.1). The regions of usefulness are in Figure (3.6). It is seen that for this example the regions do not change drastically when the identified system tensors are used to calculate control tensors.

IV. CONCLUSIONS

The authors believe that this paper contains the first published report of numerical solutions to the problem of computing nonlinear optimal feedback controls with tensors. Though the numerical bookkeeping with tensors does require a substantial initial software investment, the control results are encouraging and apply in quite general situations.

V. REFERENCES

1. J.A. O'Sullivan, "The Computer Implementation of the Tensor Solution to the Nonlinear Optimal Control Problem", M.S.E.E. Thesis, Dept. of Electrical Engineering, University of Notre Dame, May 1984.
2. M.R. Buric, "Optimal Regulation of Nonlinear Systems by Methods of Tensor Algebra", Ph.D. Dissertation, Dept. of Electrical Engineering, University of Minnesota, January 1978.
3. M.R. Buric, "Application of Multilinear Algebra in Optimal Regulation of Nonlinear Polynomial Systems", Proc. 16th Allerton Conference on Comm., Control, and Computing, Urbana, Illinois, University of Illinois, 1978.
4. E.G. Al'brekht, "On the Optimal Stabilization of Nonlinear Systems", J. Appl. Math. Mech., Vol. 25, pp. 1254-1266, 1961.
5. W.L. Garrard, N.H. McClamroch and L.G. Clark, "An Approach to Suboptimal Feedback Control of Non-linear Systems", Int. J. Control, Vol. 5, pp. 425-435, 1967.
6. W.L. Garrard, "Suboptimal Feedback Control for Nonlinear Systems", Automatica, Vol. 8, pp. 219-221, 1975.
7. D.L. Lukes, "Optimal Regulation of Nonlinear Dynamical Systems", SIAM J. Control, Vol. 7, pp. 75-100, 1969.
8. W. Greub, Linear Algebra, Fourth Edition. New York: Springer-Verlag, 1981.
9. W. Greub, Multilinear Algebra, Second Edition. New York: Springer-Verlag, 1978.
10. W.L. Garrard and J.M. Jordan, "Design of Nonlinear Automatic Flight Control Systems", Automatica, Vol. 13, pp. 497-505, 1977.
11. J.P. Hill, "Solution of Nonlinear Optimal Control Problems Using the Algebraic Tensor: An Example", M.S.E.E. Thesis, Dept. of Electrical Engineering, University of Notre Dame, May 1982.
12. S. Yurkovich, "Application of Tensor Ideas to Nonlinear Modeling and Control", M.S.E.E. Thesis, Dept. of Electrical Engineering, University of Notre Dame, January 1981.
13. S. Yurkovich, "Multilinear Modeling and Linear Analysis of Nonlinear Systems", Ph.D. Dissertation, Dept. of Electrical Engineering, University of Notre Dame, May 1984.
14. D. Bugajski, "Monomial Reduction of Nonlinear Tensor Models", M.S.E.E. Thesis, Dept. of Electrical Engineering, University of Notre Dame, May 1984.
15. L. McWilliams, "Nonlinear Models by Layered Tensors", M.S.E.E. Thesis, Dept. of Electrical Engineering, University of Notre Dame, November 1984.
16. R.W. Brockett, "Nonlinear Systems and Differential Geometry", Proc. of IEEE, Vol. 64, pp. 61-72, 1976.

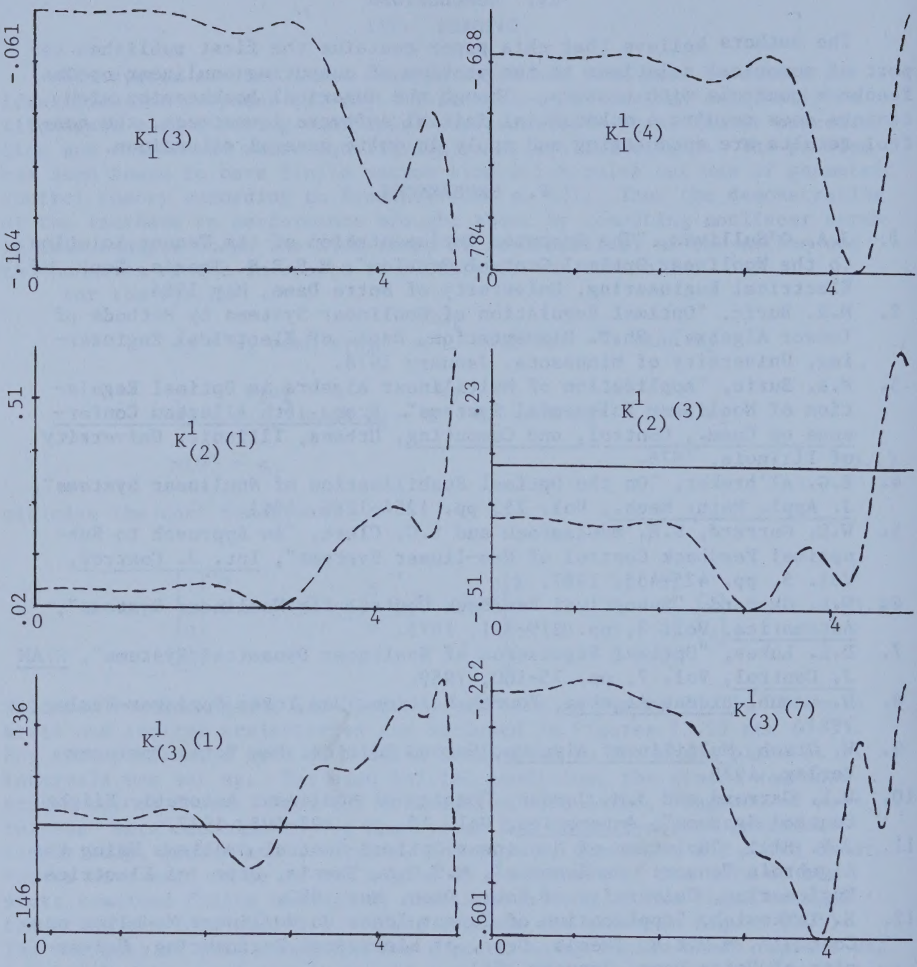


Figure 3.1 Selected Feedback Gains. Note that the linear feedback is

$$u(1) = K_1^1(1) * x(1) + K_1^1(2) * x(2)$$

$$u(2) = K_1^1(3) * x(1) + K_1^1(4) * x(2)$$

and the quadratic part of the feedback is

$$u(1) = K_{(2)}^1(1) * x(1) **2 + K_{(2)}^1(2) * x(1) * x(2) + K_{(2)}^1(3) * x(2) **2$$

$$u(2) = K_{(2)}^1(4) * x(1) **2 + K_{(2)}^1(5) * x(1) * x(2) + K_{(2)}^1(6) * x(2) **2.$$

The third order part of the feedback (omitted due to space limitations) is calculated in an analogous manner.

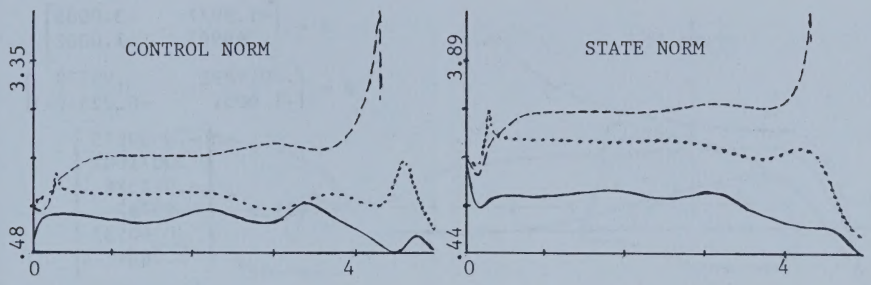


Figure 3.2a) Comparative control and state norm trajectories versus time for x(0) = (2.0,0.7) (Linear goes unstable)

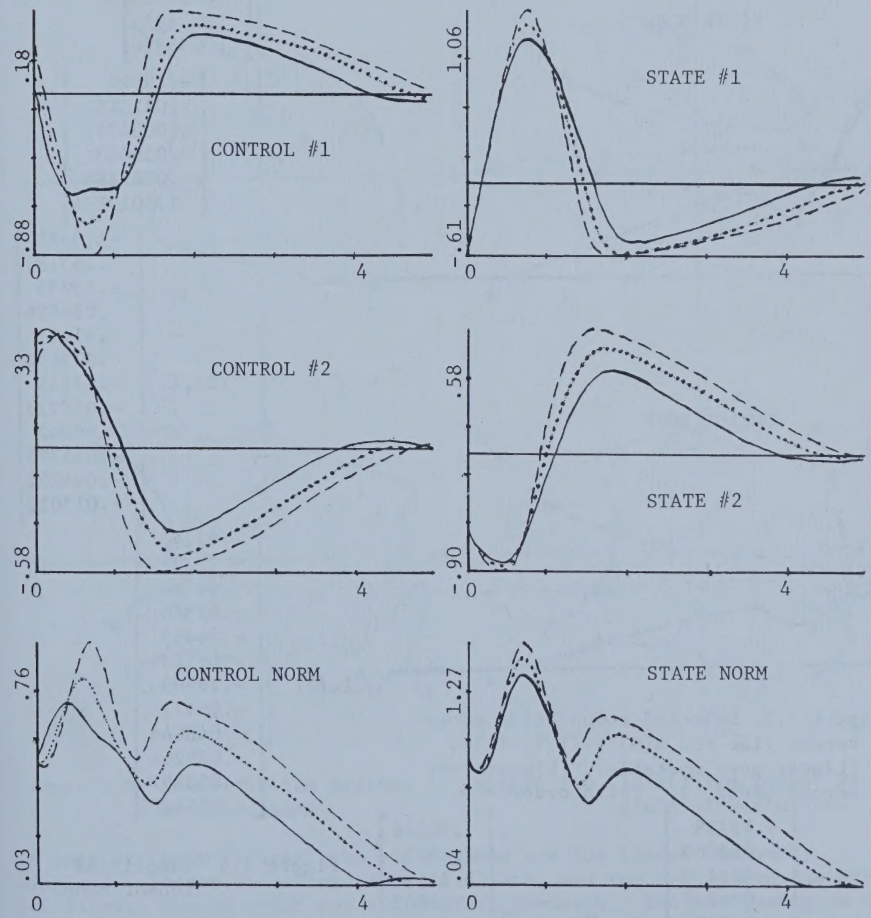


Figure 3.2b) Plots versus time for x(0) = (-.6,-.6). For all plots, the legend is linear ----, second order ...., third order ——.

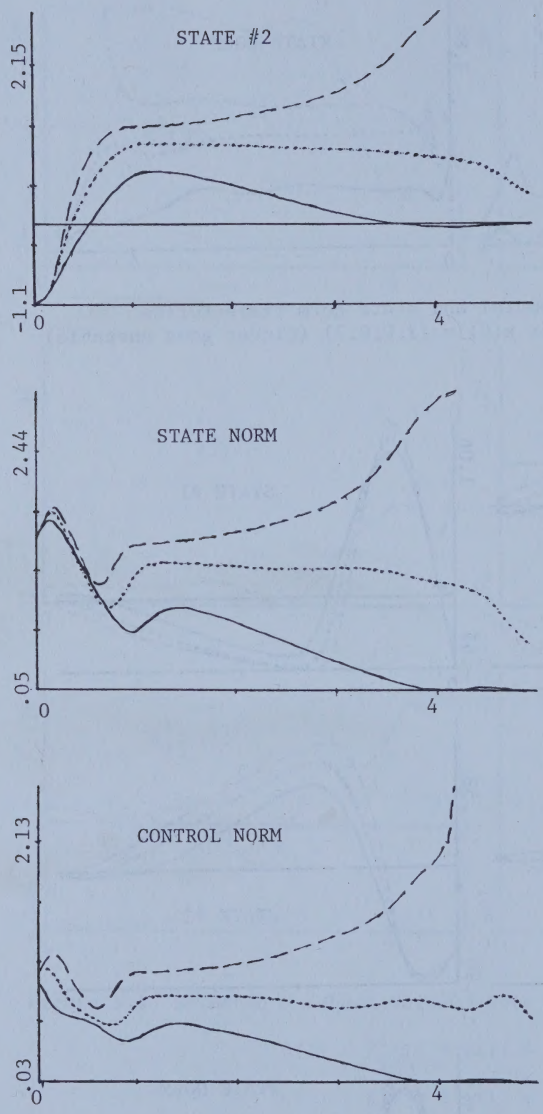


Figure 3.3 Selected comparative plots versus time for x(0) = (1.1,-1.1). (Linear goes unstable). Linear ---, second order ..., third order ——.

$$A = \begin{bmatrix} -1.9977 & -3.0008 \\ .99967 & 1.0002 \end{bmatrix}$$

$$B = \begin{bmatrix} .0019891 & .99779 \\ -1.0001 & -6.2231E\text{-}7 \end{bmatrix}$$

$$A_{(2),0} = \begin{bmatrix} -.0039675 \\ .0071601 \\ -.012386 \\ -.99787 \\ .0010537 \\ 7.4681E\text{-}4 \end{bmatrix}$$

$$A_{1,1} = \begin{bmatrix} -4.0692 \\ .027909 \\ .026129 \\ -.0078805 \\ .021083 \\ .0026825 \\ .002591 \\ .0035398 \end{bmatrix}$$

$$A_{0,(2)} = \begin{bmatrix} -.19024 \\ -.012365 \\ -.0034771 \\ .036033 \\ .0022184 \\ 1.5017E\text{-}4 \end{bmatrix}$$

$$A_{(2),1} = \begin{bmatrix} -3.5488 \\ .16518 \\ -.10235 \\ .036896 \\ .41411 \\ .075615 \\ -.17155 \\ -.055923 \\ .96002 \\ -.035393 \\ -.044252 \\ -.013022 \end{bmatrix}$$

$$A_{1,(2)} = \begin{bmatrix} .62148 \\ -.25223 \\ .44724 \\ -.61366 \\ -.76455 \\ -.67544 \\ -.28909 \\ -.05473 \\ -.048646 \\ -.025264 \\ -.058202 \\ -.0056896 \end{bmatrix}$$

Figure 3.4 Identified Tensor Model

$$A_{(3),0} = \begin{bmatrix} -.18738 \\ -.032069 \\ -.0251 \\ -.019747 \\ -.51749 \\ -.016328 \\ -.019048 \\ -.0011802 \end{bmatrix} \qquad A_{0,(3)} = \begin{bmatrix} .70548 \\ 1.6627 \\ 3.3163 \\ .78087 \\ -.2223 \\ -.6684 \\ -1.2624 \\ -.3395 \end{bmatrix}$$

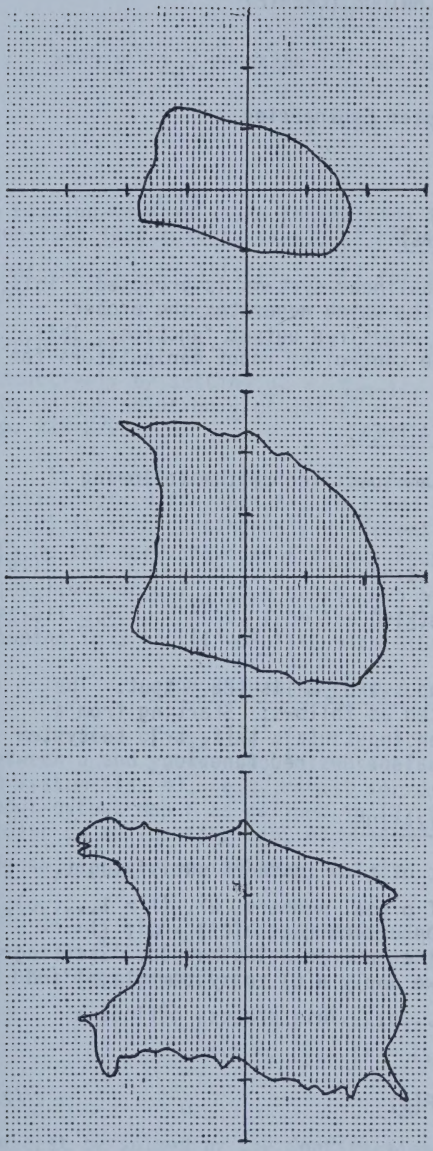


Figure 3.5 Actual System Regions of Usefulness

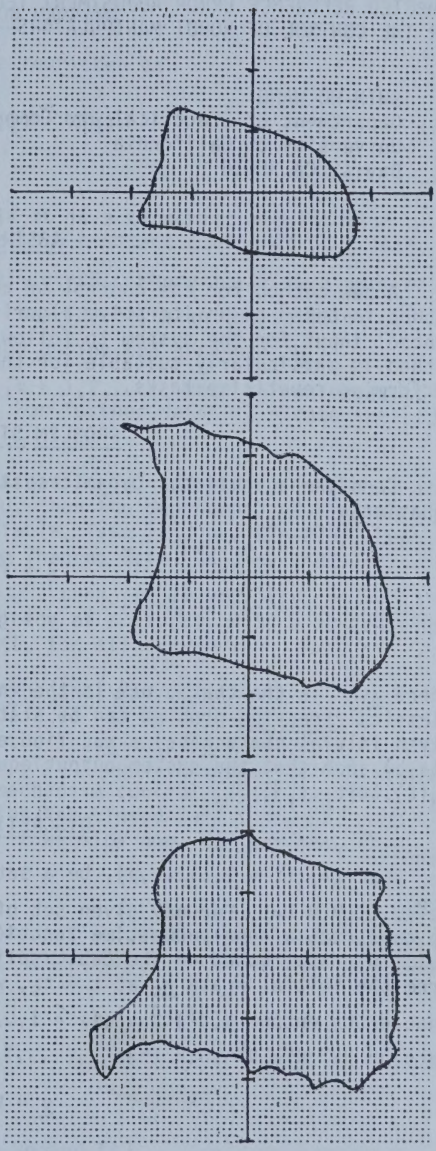


Figure 3.6 Identified Tensor Regions of Usefulness

In both of these figures, the top regions are for linear feedback, middle for linear and second order feedback, and the bottom regions are for linear, second order and third order feedback. The hash marks on the axes are at 1.0 intervals. x_1 is the horizontal axis.

COMPUTED REFERENCE ERROR ADJUSTMENT TECHNIQUE (CREATE) FOR THE CONTROL OF ROBOT MANIPULATORS

SHIH-HO WANG
Department of Electrical and Computer Engineering
University of California
Davis, CA 95616

ABSTRACT

A new method for the control of nonlinear systems is proposed. This method is based on an iterative algorithm which generates the proper control input for the system to follow a desired output function. This method is particularly suitable for the control of robot manipulators for performing repetitive tasks.

PROBLEM STATEMENT

Consider a nonlinear dynamic system specified by the following equations:

$$\dot{\underline{x}} = f(\underline{x},u) \tag{1a}$$

$$y = g(\underline{x},u) \tag{1b}$$

where $\underline{x}\epsilon R^n$ is the state, $u\epsilon R$ is the input and $y\epsilon R$ is the output. The classical tracking problem can be stated as follows: Starting from a given initial state $\underline{x}(t_s) = \underline{x}_s$, find an input $u(t)$, $t\epsilon[t_s,t_f]$, to system (1a,b) such that the system output equals a desired function $y_d(t)$,$t\epsilon[t_s,t_f]$.

DISCRETIZATION

Let t^i, $(i = 1,.....,p)$, be a set of time instants satisfying $t_s = t^1 < t^2 < ... < t^p = t_f$. For any given set of p real numbers, u^i, $(i = 1,...,p)$, we construct a continuous-time function u such that $u(t^i) = u^i$, $(i = 1,...,p)$, say, based on the cubic spline interpolation method. Starting from the initial state $\underline{x}_s$ at $t = t_s$, we apply $u(t)$ to system (1a,b) and record the output $y(t^i) \triangleq y^i$,$(i = 1,...,p)$. This process defines a mapping $F: R^p \rightarrow R^p$ such that

$$\underline{y} = F(\underline{u}) \tag{2}$$

where $\underline{y} = (y^1,...,y^p)'$ and $\underline{u} = (u^1,...,u^p)'$.

The original continuous-time tracking problem can be reformulated as the problem of solving nonlinear equation (2) when $\underline{y}$ equals a given value $\underline{y}_d\epsilon R^p$.

CONTRACTION MAPPING

When $\underline{y} = \underline{y}_d$, equation (2) can be rewritten as

$$\underline{u} = \underline{u} + \underline{y}_d - F(\underline{u}) \triangleq K(\underline{u}). \tag{3}$$

A mapping $K: R^p \to R^p$ is a contraction if it is a Lipschitz mapping with constant less than 1, (i.e., if there is a constant C with $0 < C < 1$ such that $||K(\underline{u}) - K(\underline{v})|| \leq C||\underline{u} - \underline{v}||$ for all $\underline{u},\underline{v} \in R^p$, where $||\cdot||$ is a norm on R^p). If K is a contraction mapping, then (3) has a unique solution $\underline{u}$. Furthermore, if we choose any $\underline{u}_0 \in R^p$ and define the sequence $\{\underline{u}_m\}_0^\infty$ inductively by setting $\underline{u}_m = K(\underline{u}_{m-1})$, $(m = 1,2,\ldots)$, then $\underline{u}_m$ converges to the unique solution $\underline{u}$ of (3) as $m \to \infty$. For further details, please see [1], pp. 228-229.

It can be shown that if the functions $f(\underline{x},u)$ and $g(\underline{x},u)$ in (1a,b) are continuously differentiable in $\underline{x}$ and u, then the dependence of $F(\underline{u})$ on $\underline{u}$ in (2) is also continuously differentiable, (see [1], pp. 274-275). Let $dK_{\underline{u}}$ be the derivative of K at $\underline{u}$. From Theorem 7.4, ([1], p. 149), if $||dK_{\underline{u}}|| \leq C$ for all $\underline{u}$ in a convex set $S \subset R^p$, then $||K(\underline{u}) - K(\underline{v})|| \leq C||\underline{u} - \underline{v}||$ for all $\underline{u},\underline{v} \in S$. Hence, from the definition of K in (3), K is a contraction mapping in S if $||I - dF_{\underline{u}}|| \leq C$ with $0 < C < 1$, for all $\underline{u} \in S$.

SERVOMECHANISM CONTROL AND CONTINUATION METHOD

Instead of dealing directly with the input-output relationship between $\underline{u}$ and $\underline{y}$, one should consider applying some feedback control laws to system (1a, b), (e.g.,robust servomechanism control [2],[3]). The input-output map F as defined in (2) for the closed-loop system is much closer to an identity map, especially if we slow down the process, (i.e., choose large value for $T \triangleq t_f - t_s$).

We may also consider choosing different values for the length of operation, T_i, $(i = 1,2,\ldots)$, where $T_1 > T_2 > \ldots > 0$. The corresponding family of nonlinear equations,

$$y = F_{T_i}(\underline{u}), \quad (i = 1,2,\ldots), \tag{2'}$$

can best be solved by the Continuation Method [4].

CONCLUSIONS

A new approach to the control of nonlinear systems is outlined above. The appropriate control input for the system to follow a given output function is found by repetitive testings on the physical systems; hence no

DESIGN OF A CONTROL SYSTEM FOR BATCH PRODUCTION ON A DISTILLATION COLUMN

M. K. Yurtseven
Purdue University
at Indianapolis, IN

W. F. Fincham
Queen Mary College
University of London, UK

ABSTRACT

The objective of this control system is to regulate the production of a fixed number of batches on a continuously operated distillation column in an optimum manner. Due to the "impossibility" of an optimum solution, the problem is decomposed into subproblems of batch sequencing, steady-state optimization and dynamic control and optimization. These subproblems are solved independently and they are then integrated to achieve the overall goal.

DESCRIPTION OF THE CONTROL SYSTEM

The decomposition procedure for this problem is based on "Multilevel-Multihorizon Production Control Techniques" described by Milkiewicz (1). Some industrial applications which make use of similar approaches are given in (2). Recent trends in process control have similar approaches to those mentioned above. The obvious difference in recent studies is the growing emphasis on microprocessor-based distributed control systems as opposed to centralized control through a single computer (3,4). The control structure developed for the column is shown in Figure 1.

Production Tasks describing the volumes and the compositions of the batches are fed into Model 1. This model is an operating economic one with a time horizon of one to four weeks. It determines the optimum steady-state operating conditions by a nonlinear mathematical programming technique. Model 2 sequences the batches using the Traveling Salesman Algorithm to minimize the cummulative charge-over cost. Model 3 is an inferential type with a time horizon of a few hours to one day. It observes that the batches are produced at the right volumes and compositions. Also, it informs the on-line controller about the set-points and the batch switching times. Model 4 includes a dynamic model of the column and an extended Kalman filter. It has a time horizon of 64 seconds, which is the sampling interval set by the interface and the on-line computer. The filter filters out the process and the measurement noise and estimates the "best" values of the state variables and some of the model parameters. The control parameters are updated by the adaptation algorithm. This is necessary for proper control since the distillation process is inherently nonlinear. Furthermore, the nonlinearity is worsened due to the wide variations in the operating conditions in the batch mode. Finally, the feedback loops between all the models ensure the stable operation of the control system.

It is possible to improve this design in two aspects. The first one is to include Model 3 in the on-line mode of operation. This requires additional instrumentation, but it can ease some of the burden placed on the operator. Secondly, a communication link between the on-line computer and the off-line computer can be established. This will automate the integration between the steady-state optimization and the sequencing problems that are handled by the off-line computer, with on-line control. Full details of this study and the simulation results for a case study can be found in (2).

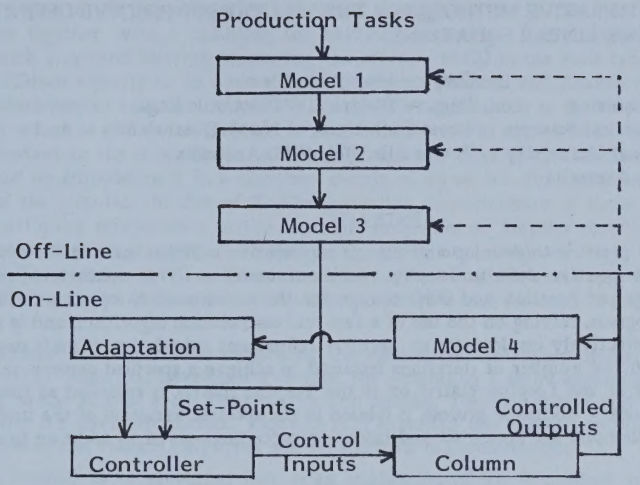


Figure 1. Control structure for the column

REFERENCES

1. Milkiewicz, F., "Multilevel-Multihorizon Production Control", Symposium on Systems Approach to Developing Countries, Algeria, May 1973.

2. Yurtseven, M.K., "Optimization of Batch Production and Control Policy on a Pilot-Scale Distillation Column", Ph.D. Thesis, University of London U.K., 1976

3. Bennet, S. and Linkeu, D.A. (Editors), "Computer Control of Industrial Processes", IEE Publication, London U.K., 1982.

4. Skrokov, M.R. (Ed.), "Mini-and Microcomputer Control in Industrial Processes", Van Nostrand 1980.

FAST AND STABLE ITERATIVE METHODS FOR THE SOLUTION OF POSITIVE DEFINITE TOEPLITZ SYSTEMS OF LINEAR EQUATIONS[1]

Robert R. Bitmead
Dept. of Systems Engineering
Research School of Physical Sciences
The Australian National University
Canberra, ACT, 2600, Australia

Gregory Hamilton Allen
Dept. of Electrical & Electronic Eng.
James Cook Univ. of North Queensland
Townsville, Qld, 4811, Australia

ABSTRACT

The aim of this paper is to develop and discuss an iterative solution method for systems of linear equations with positive definite Toeplitz coefficient matrices. This method requires O(N LOG(N)) computations per iteration and O(N) storage for the solution of N equations. The algorithm is simple to program, relying on the use of a fast real convolution algorithm, and is numerically stable since it effectively implements an iterative refinement calculation at each step. Convergence is linear and the number of iterations required to achieve a specified error is related to the condition number of the Toeplitz matrix or, if the Toeplitz matrix is regarded as the covariance matrix of a stationary stochastic process, is related to the spectral variation of the underlying process. Numerical examples are presented and inversion procedures given in addition to solution methods.

INTRODUCTION

A Toeplitz matrix T has its ijth element a function of i-j, i.e.

$$(T)_{ij} = T_{i-j}. \tag{1}$$

One important feature of such matrices, which we shall use to considerable advantage later, is that they are completely specified by the elements of their first row and first column since the rest of the matrix may be generated by propagating these elements parallel to the diagonal.

Systems of linear equations with Toeplitz coefficient matrices arise in many applications such as time series analysis [1], spectral analysis [2], seismic deconvolution [3], ionospheric transmission, digital filtering [5], numerical analysis, beamforming, etc.

Consequently, there is considerable interest in developing computational methods for the inversion of Toeplitz matrices which utilize their inherent structure. More specifically, one usually attempts to solve the equations

$$Tx = y \tag{2}$$

for x given T and y, so that explicit computation of the inverse of T is not necessary.

In very many of the applications, the Toeplitz matrix T arises as a covariance matrix (or matrix of lagged correlations) and so possesses still further structure since it is symmetric positive definite. In these applications one frequently also has a special form of (2) where the elements of y are also composed of correlation data. These are the Normal or Yule-Walker equations, although we do not consider them in particular here. Our aim here is to consider positive definite T and to develop and discuss an iterative method for the solution of (2) and for the construction of T^{-1} which is fast and numerically stable. Since the dominant feature of Toeplitz matrix applications is that very often the dimension of T, n+1, is very large, computational and storage efficiency is of great concern. The history of the development of Toeplitz matrix inversion methods has been one of improvement in the operation count and memory requirements. Our method will cost O(n log n) in computations (additions and multiplications) per iteration and O(n) in storage.

The 'standard' algorithm for the inversion of Toeplitz matrices is due to Levinson [6] which generates a Cholesky decomposition of T. Many variants and improvements have been made to this basic algorithm by Trench [7], Zohar [8], Bareiss [9], Whittle [10], Durbin [11] Burg [12], Akaike [13] and others. Further, Cybenko [14] has demonstrated that the algorithm is numerically stable when applied to positive definite T. The Levinson algorithm has a computational complexity of $O(n^2)$ for the generation of T^{-1} as compared to standard gaussian methods $O(n^3)$. These aspects of computational saving and numerical stability account for the standardization on this algorithm by users such as the oil exploration industry.

More recently several people have presented methods for the solution of (2) which are computationally more efficient than the Levinson algorithm. In particular, these schemes have an

operation count of O(n log n log n) which arises through the use of fast Fourier transform (FFT) techniques together with a doubling (or halving) operation. Brent, Gustavson and Yun [15] develop such a method through connecting the solution to (2) to the Pade table computation then to the Euclidean algorithm. In a more direct procedure Bitmead and Anderson [16] and Morf [17] use the displacement rank results of [18,19] and halving methods to develop an O(n log n log n) algorithm which is still quite involved to program and would appear to have a very large multiplying constant in the order above. Kumar [20] has also developed a solution method of similar speed based on embedding T in a circulant matrix of twice the dimension and then relating the inverse of the circulant to that of T. The pervading characteristics of these procedures are that they are all quite complicated, utilize algebraic properties of Toeplitz matrices and/or their elements, make few concessions to or statements about numerical accuracy, and are effectively untested. These methods all are concerned with generating an exact solution for (2). In use, numerical errors arise and iterative refinement techniques may be used to improve the numerical solution.

A number of people have considered the development of solely iterative methods for the solution of (2). Rino [21] developed a method for generating a series expansion solution to (2) by writing T as the sum of a circulant matrix and another Toeplitz matrix and presented a method for choosing the circulant matrix. Ekstrom [22] extended this method to block Toeplitz systems and Silverman and Pearson [23] apply similar methods to deconvolution.

The method to be presented here is an application of Rino's method to positive definite T. The procedure that we derive will be shown to be guaranteed geometrically convergent and numerically stable. Our contribution is in the specification of a simply computed, guaranteed effective circulant matrix for the series expansion. This effects considerable computational saving and makes the basic approach feasible. The structure of the paper is as follows. In the next section we present and discuss Rino's method and develop our procedure. Section 3 is devoted to a discussion of the properties of the algorithm, especially with regard to convergence rate, and Section 4 contains some examples illustrating the performance.

THE ITERATIVE SOLUTION METHOD

A circulant matrix C is a special form of Toeplitz matrix whose rows are a circular shift of the previous row.

$$C_{ij} = C_{(i-j) \bmod (n+1)} \tag{3}$$

The inverse of a circulant matrix is easily computed using the FFT and is again a circulant matrix [24].

The Toeplitz inversion procedure proposed by Rino is to write

$$T = C + W \tag{4}$$

where C is circulant and, of course, W is Toeplitz. C^{-1} may be found by FFT and we write

$$T = C(I + C^{-1}W) = C(I + E) \tag{5}$$

where

$$E = C^{-1}W. \tag{6}$$

T^{-1} is thus of the form

$$T^{-1} = (I + E)^{-1}C^{-1} \tag{7}$$

which has the series expansion in powers of E

$$T^{-1} = (I - E + E^2 - E^3 + \cdots)C^{-1} \tag{8}$$

In order for this series to converge it is necessary and sufficient that

$$|\lambda_i(E)| < 1 \tag{9}$$

where λ_i are the eigenvalues. In this instance, the solution x to (2) may be evaluated by the recursion

$$x_1 = C^{-1}y; x_{k+1} = x_1 - Ex_k \tag{10}$$

We will leave in abeyance the computational questions until the next section but consider here the crucial question of the satisfaction of (9).

Since E is defined by (6) and C and W by (4) the problem of satisfying (9) is akin to choosing a circulant matrix which is close to a given Toeplitz matrix. The procedure developed in [21] is to choose

$$c_i = \sum_{m=-\infty}^{\infty} t_{(i+mn)} \tag{11}$$

where $t_i = 0$ for $|i| > n$. For large values of n this C approaches T when the first row and column of T are constrained to be absolutely summable, Gray [25]. The asymptotic equivalence of C and T as $n \to \infty$ is used to argue that (9) will be satisfied for finite n although one can easily construct examples for which this does not work, mostly for small n.

For example consider the symmetric Toeplitz matrix

$$T = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}$$

With $|\alpha| < 1$, T is positive definite. This leads to

$$C = \begin{bmatrix} 1 & 2\alpha \\ 2\alpha & 1 \end{bmatrix} \quad \text{and} \quad E = \frac{1}{(1-4\alpha^2)} \begin{bmatrix} 1 & -2\alpha \\ -2\alpha & 1 \end{bmatrix}$$

which shows that for $\alpha = 1/2$ C is singular while for $\alpha > 1/2$ E violates (9). We shall have more to say concerning this problem later.

To overcome this reliance on asymptotic theory for the satisfaction of the convergence criterion (9) we propose the choice of circulant matrix

$$C = \gamma I \tag{12}$$

where

$$\gamma > \frac{1}{2} \lambda_{\max}(T) \tag{13}$$

Clearly, C of (12) is readily inverted without recourse to FFT computation. Following this we have

$$E = C^{-1}(T - C) = \gamma^{-1}T - I \tag{14}$$

and the bound (13) ensures that (9) holds while the structure of C causes E to remain symmetric and Toeplitz (This will come in as important later in maintaining a low operation count).

Because E is symmetric Toeplitz defined by (14), E has real eigenvalues and it is easy to show that

$$\lambda_{\min}(E) = \gamma^{-1} \lambda_{\min}(T) - 1 \tag{15}$$

$$\lambda_{\max}(E) = \gamma^{-1} \lambda_{\max}(T) - 1 \tag{16}$$

The geometric convergence rate of the iterative algorithm (10) is determined as the maximum modulus of $\lambda\min(E)$ and $\lambda\max(E)$. The positive definiteness of T and (13) guarantee that these moduli are less than one. If one were to optimize this convergence rate by minimizing the maximum modulus eigenvalue of E, then one arrives at the optimal value of

$$\gamma_{opt} = 1/2(\lambda_{\max}(T) + \lambda_{\min}(T)) \tag{17}$$

Further it becomes apparent that the condition number of m arises as important in determining the best achievable convergence rate.

The choice of γ equal to a fixed multiple of $\lambda\max(T)$ would be numerically prohibitive and so for the application of this method we need to overbound $\lambda\max(T)$ in a computationally cheap manner. It is well known [15] that the maximum row sum norm of a matrix overbounds its spectral norm, i.e. here

$$\max_{i=1,\cdots,n+1} \sum_{j=1}^{n+1} |(T)_{ij}| \geqslant \lambda_{\max}(T) \tag{18}$$

and this may be used to generate a suitable value of γ for (12).

In brief, the new algorithm for solving (1) involves

(i) selecting positive constant γ satisfying (13) using (18);

(ii) computing E according to (14) and x according to (10); and

(iii) iterating (10).

In the next section we go on to consider practicalities of the implementation of the procedure and to quantify performance. The novelty of the approach here is to devise a method which is guaranteed convergent and numerically stable.

IMPLEMENTATION

Fast Methods

Since our aim here is to use the structure of the Toeplitz matrix T to save on computational complexity and storage requirements we shall now investigate the savings of the new scheme.

The first task is to compute an overbound for $\lambda\max(T)$ using (18). Because T is Toeplitz, this is achievable in 2n additions as follows. Compute the sum of moduli of elements of the first row of T, n additions. Next generate the second row sum by adding $|t_1|$ to and subtracting $|t_n|$ from the previous sum. Continue in this way for the remaining first half of the rows (T is symmetric so there is no need to check the second half of the rows), keeping the maximum value, n additions. The parameter γ may now be generated as a multiple of this maximum row sum.

To iterate (10) one first needs to calculate x_1 and E. Equations (12) and (14) show that

$$x_1 = \gamma^{-1}y \tag{19}$$

which involves (n + 1) multiplications, and that E is symmetric and Toeplitz with first column

$$e_i = \gamma^{-1}t_i - \delta_{io}, \quad i = 0,1,2,\cdots,n$$

again this requires O(n) computations. With regard to storage requirements it is apparent that x_1 uses n + 1 values as does E since it is symmetric and Toeplitz and we only need to store its first row or column.

Once x_1 and E are computed the iteration of (10) necessitates the evaluation of the product Ex_k. For generic matrices E this would involve $O(n^2)$ computations. However, since E is Toeplitz

$$(Ex_k)_i = \sum_{j=1}^{n+1} e_{i-j}(x_k)_j \tag{20}$$

That is, the elements of this product are some of the terms in the convolution of $[x_{k_1}x_{k_2}\cdots x_{k_{n+1}}0\cdots 0]$ and $[e_{-n}e_{-n+1}\cdots e_{-1}e_0e_1\cdots e_n]$. There are a number of fast convolution procedures which can be used to perform this real convolution, see e.g. [26], all of which operate in O(n log n) computations.

The total computational cost of the implementation of the solution method is thus O(n) for the initial evaluation of E and x_1, and O(n log n) per iteration of the algorithm (10). We shall address the next question of convergence rate and number of iterations shortly, but firstly it is worthwhile to consider a doubling algorithm for the computation of (8), in order to highlight the numerical stability of the above method.

A Doubling Method

In attempting to derive faster procedures for solving (2) using

$$x = (I - E + E^2 - E^3 + \cdots)C^{-1}y \tag{21}$$

it is possible to use the following approach

$$x_1 = C^{-1}y;\ x_2 = x_1 - Ex_1,\ E^2 = EE;\ x_4 = x_2 + E^2x_2,\ E^4 = E^2E^2; \tag{22}$$

However, since there is no obviously cheap form of storage for E^{2^N} or for its multiplication of a vector, as E^{2^N} is no longer necessarily Toeplitz, these multiplications by powers of E require many successive convolutions for their evaluation. Consequently, there is no computational advantage in such a doubling algorithm.

Numerical Stability

The iteration (10) is guaranteed convergent by the satisfaction of (9). The accuracy of this convergence and its sensitivity to computational or roundoff errors is enchanced by the fact that (10) has a single fixed point $x = (I - E)^{-1}x_1$ which is the correct solution provided x_1 and E are computed accurately. The iterative method is thus very robust and particularly numerically stable. This should be compared with (22) which clearly accumulates arithmetic errors and so is

not numerically stable. This claim of numerical stability has been borne out in examples where the solution is monotonically convergent to the correct value until the machine accuracy limit is reached after which the solution remains within this neighborhood of the correct value.

It is worthwhile reiterating that the approach which we are advocating is effectively a variant on the iterative refinement techniques frequently used to correct solutions of inversion algorithms which are not sufficiently accurate [27]. Consequently, it comes as no surprise that our algorithm is stable and accurate.

Convergence Rate Quantification

The solution algorithm is geometrically convergent with a rate dependent upon the maximum modulus eigenvalue of the matrix E. Recall that the extremal eigenvalues of E are given by (15) and (16), and that it is possible to define an optimal value of by (17) to maximize the convergence rate. It is apparent that the actual convergence rate will be determined by two effects; firstly, the ability to choose the parameter close to $\lambda\max(T)$ and, secondly, the condition number of T, i.e. the ratio of $\lambda\min(T)/\lambda\max(T)$.

There are very few results relating the different norms of a matrix, so that in general we are unable to guarantee the closeness or otherwise of γ to λ max(T). The important requirement of (13) is, however, guaranteed. With regard to evaluating the condition number of a positive definite Toeplitz matrix T we may appeal to the results of [24, 25] where it is shown that asymptotically the eigenvalues of T approach those of the related circulant matrix C as the dimension of T increase, while the absolute summability of the first row is maintained. The eigenvalues of a circulant matrix with elements as in (3) are given by the function

$$c(z) = \sum_{i=o}^{n} c_i z^i$$

evaluated at $x = \exp(j2\pi k / n+1)$, k = 1,2,...,n.

The conclusion of this argument is that if T is generated as the nxn covariance matrix of a stationary random process, ν_k, then as n increases, the eigenvalues of T approach the values of the power spectral density of ν_k at equispaced points around the unit circle. In particular, the condition number of T is approximately determined by the variation on the unit circle of the spectrum of the process generating T. We should say here that Rino [21] and Ekstrom [22] both give simple tests for estimating the actual convergence rate by comparing norms of differences of iterates.

Banded Toeplitz Matrices and Block Toeplitz Matrices

In many applications positive definite Toeplitz matrices arise which are banded, i.e. $t_i = 0$ $|i| > m$ for some $m < n$, see [28]. In these situations fast variants of the Levinson algorithm may be used to take advantage of the low value of m, [28]. Here the iterative algorithm may be applied as before except that now we have a banded E matrix and the iterations involve lower order convolutions. This permits an improvement in speed for these applications and represents a simple adjustment to the regular procedure.

For multichannel estimation problems block Toeplitz matrix equations arise where T is composed of elements $(T)_{ij}$ which are themselves matrices which may or may not also be Toeplitz [22, 29]. Again the algorithm does not alter since the Toeplitz nature is important only in reducing computations for γ and for fast convolution methods for multiplying a vector by E. A brief analysis shows that these procedures carry over to the block Toeplitz case and in the doubly Toeplitz case (i.e. with Toeplitz blocks) admit still further savings as in [22].

Inversion vs Solution

In some instances it may be beneficial to generate the solutions to (2) for many different y vectors and one may be interested in constructing T^{-1} rather than solving for x at each distinct time. This raises two related questions; can the methods presented here be used to produce T^{-1} efficiently, and can T^{-1} be stored efficiently so that it may be rapidly multiplied by a given y-vector? The answer to these questions is affirmative and we shall now show how the results of Gohberg and Semencul [30] may be utilized to computational advantage here.

Firstly we consider the solution of the equation $Tx_1 = y_1$ with

$$y_1 = [1 \quad 0 \quad \cdots \quad 0]^t \tag{23}$$

defining

$$x_1 = [\alpha_0 \quad \alpha_1 \quad \cdots \quad \alpha_n]^t \tag{24}$$

If $\alpha_p \neq 0$ then this allows the definition

$$a_1 = \alpha_1 / \alpha_0 \quad i = 0,1,\cdots,n$$

and T^{-1} may be written as

$$T^{-1}=\alpha_0 \left(\begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_1 & 1 & \cdot & 0 \\ \cdot & & \cdot & \cdot \\ \cdot & & & \cdot \\ a_n & \cdots & a_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & a_1 & & a_n \\ 0 & 1 & \cdot & \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & a_1 \\ 0 & \cdots & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & & & 0 \\ a_n & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ a_1 & \cdots & a_n & 0 \end{bmatrix} \begin{bmatrix} 0 & a_n & \cdots & a_1 \\ \cdot & & & \\ \cdot & & \cdot & \\ \cdot & & & a_n \\ 0 & \cdots & 0 & \end{bmatrix} \right) \tag{25}$$

where these matrices in (25) are triangular and Toeplitz. (For the more general application to block Toeplitz matrices or when $\alpha_0 = 0$ see [31, 32], although the extension here is straightforward.)

The solution of (2) with y given by (23) is the task of the algorithm of this paper. The vector x_1 is thus fast to compute, O(n log n), and the factored form of T^{-1} in (25) is easily stored by just retaining α_0 and the a_i. The remaining question of efficiency of multiplication of a given vector by T^{-1} is also resolved by keeping the factored structure of (25). This latter step is achieved rapidly by utilizing the earlier FFT convolution results for effecting the multiplication of a vector by a Toeplitz matrix.

Thus it appears that the derivation of T^{-1} requires only marginally more work than a single solution procedure, requires the same storage, but need not necessarily be as numerically stable since it conceivably could involve differencing of large numbers.

EXAMPLES

We have presented the fast iterative algorithm for the solution of positive definite Toeplitz systems of linear equations. In this section we shall present computed examples demonstrating the various claims of the previous section concerned with convergence rate and numerical stability. Throughout this section we shall use positive definite Toeplitz matrices T which are the covariance matrices of an AR(2) stationary random process and our benchmark examples will be to solve the Yule-Walker or Normal equations, since in this instance a closed-form solution is known. Also it is straightforward to generate covariances of processes with different power spectral densities by adjusting the parameters, in order to show the change in convergence rates.

The positive definite Toeplitz matrices of these examples are the covariance matrices of AR(2) stationary stochastic processes and are generated by the recurrence relation

$$t_o = 1 \ , \ t_1 = \frac{\phi_1}{1-\phi_2}$$

$$t_k = \phi_1 t_{k-1} + \phi_2 t_{k-2} \text{ for } k \geq 2$$

where ϕ_1 and ϕ_2 are the parameters of the AR(2) process [33]. We solve the Normal or Yule-Walker equations using these $\{t_i\}$ since these equations have a well known closed form solution to which we may easily compare the iterates of the algorithm.

The AR(2) process is stable for ϕ_1 and ϕ_2 in the interior of the triangle in Figure 1. The contours inside the triangle indicate the energy amplification of the process with higher values leading to larger condition number of the constructed Toeplitz matrix. Evaluation of the power spectral density of the process shows also that the spectrum variation is similarly related to high condition number and location of the parameters within the triangle.

The subsequent figures contain plots of the mean square error, i.e. squared error divided by dimension of the vector, versus iteration number for order 10, 20, 30, 40 and 50 matrices for the solution procedure applied to the Yule-Walker equation. Notice the convergence rate is effectively independent of dimension and note the effect of spectrum variation/condition number/ϕ_1,ϕ_2-location on the rates. Examples of dimensions up to 2048 showed precisely the same effects demonstrating the suitability of the method for large computation and its numerical stability.

CONCLUSIONS

We have presented and demonstrated an iterative procedure for solving positive definite Toeplitz sets of linear equations and for generating the inverses of such Toeplitz matrices. The scheme relies on a decomposition of the Toeplitz matrix as the sum of a circulant matrix and a Toeplitz residual and then developing the solution as a power series. The technique is a variant of iterative improvement methods and so is numerically stable. By appropriate choice of circulant and use of fast convolution methods the iterates may be computed in O(n log n) operations thus producing a fast algorithm.

The algorithm is very easily programmed and is demonstrated to be geometrically convergent with a convergence rate which is related to the condition number of the matrix involved. For large Toeplitz matrices this rate is also related to the variation of the power spectral density of the process whose autocorrelation matrix is the particular positive definite matrix. This convergence rate may be estimated by comparing successive iterates on-line and thus a stopping rate implemented. The most pleasing novelty of this solution procedure is its simplicity.

REFERENCES

[1] E. Parzen, *Time Series Analysis Papers*, Holden-Day, Inc., San Francisco, California, 1967.

[2] S. M. Kay and S. L. Maple Jr., "Spectrum Analysis - a Modern Perspective", *Proc. IEEE*, vol. 69, Nov. 1981, pp. 1380-1419.

[3] R. A. Wiggins and E. A. Robinson, "Recursive solution to the multichannel filtering problem", *J. Geophys. Res.*, vol. 70, 1965, pp. 1885-1891.

[4] J. Makhoul, "Stable and efficient lattice methods for linear prediction", *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-25, Oct. 1977, pp. 423-428.

[5] J. Makhoul, "Linear Prediction: A Tutorial Review", *Proc. IEEE*, vol. 63, pp. 561-580, Apr., 1975.

[6] N. Levinson, "The Weiner RMS (Root-Mean Square) error criterion in filter design and prediction", *J. Math. and Phys.*, vol. 25, pp. 261-278, 1947.

[7] W. F. Trench, "An algorithm for the inversion of finite Toeplitz matrices", *J. SIAM*, vol. 12, pp. 515-522, Sept. 1964.

[8] S. Zohar, "The algorithm of W. F. Trench", *J. Assoc. Comput. Mach.*, vol. 16, pp. 592-601, Oct. 1969.

[9] E. H. Bareiss, "Numerical solution of linear equations with Toeplitz matrices", *Numer. Math.*, vol. 13, pp 404-424, Oct. 1969.

[10] P. Whittle, "On the fitting of multivariable autoregressions and the approximate canonical factorization of a spectral density matrix", *Biometrika*, vol. 50, pp. 129-134, 1963.

[11] J. Durbin, "The fitting of time series models", *Rev. Internat. Inst. Statist.*, vol. 28, pp 233-244, 1949.

[12] J. P. Burg, "Maximum entropy spectral analysis", PhD dissertation, Dept. Geophysics, Stanford University, Stanford, CA, May 1974.

[13] H. Akaike, "Block Toeplitz matrix inversion", *SIAM J. Appl. Math.*, vol 24, pp. 234-241, Mar. 1973.

[14] G. Cybenko, "Round-off error propagation in Durbin's, Levinson's and Trench's algorithms" in *Proceedings of the Int. Conf. on Acoustics Speech and Signal Processing*, Washington, D.C., Apr. 1979.

[15] R. P. Brent, F. G. Gustavson and D. Y. Yun, "Fast solution of Toeplitz systems of equations and computation of Pade approximants", *J. Algorithms*, vol. 1, pp. 259-295, 1980.

[16] R. R. Bitmead and B. D. O. Anderson, "Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations", *Linear Algebra and its Applications*, vol. 34, pp 103-116, 1980.

[17] M. Morf, "Doubling algorithms for Toeplitz and related equations", 1980 ICASSP 954-959, Denver, April 1980.

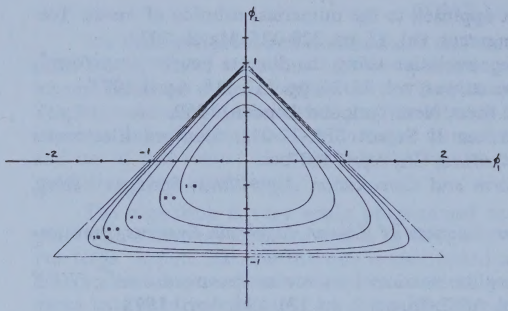
[18] B. Friedlander, M. Morf, T. Kailath, L. Ljung, "New inversion formulas for matrices classified in terms of their distance from Toeplitz matrices", *IEEE Trans. Circuits and Systems*, vol. CAS-26, no. 9, pp. 750-755, Sept. 1979.

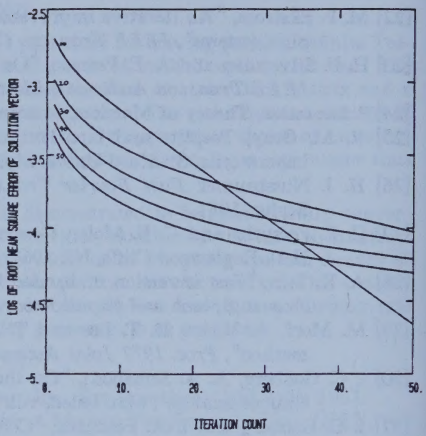
[19] T. Kailath, S.-Y. Kung and M. Morf, "Displacement ranks of matrices and linear equations", *J. Math. Analysis and Applns*, vol. 68, no. 2, pp. 395-407, Apr. 1979.

[20] R. Kumar, "A fast algorithm for solving Toeplitz system of equations", (submitted for publication).

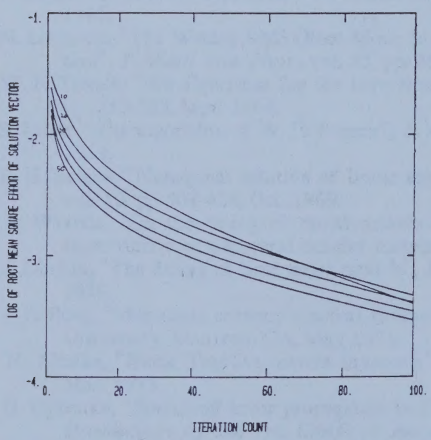
[21] C. L. Rino, "The inversion of covariance matrices by finite Fourier transformations", *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 230-232, Mar. 1970.

[22] M. P. Ekstrom, "An iterative improvement approach to the numerical solution of vector Toeplitz systems", *IEEE Trans. on Computers*, vol. 23, pp. 320-325, March 1974.
[23] H. F. Silverman and A. E. Pearson, "On deconvolution using the discrete Fourier transform", *IEEE Trans. on Audio and Electroacoustics*, vol. AU21, pp. 112-118, April 1973.
[24] P. Lancaster, Theory of Matrices, Academic Press, New York and London, 1969.
[25] R. M. Gray, Toeplitz and circulant matrices: II Report SEL-77-011, Stanford Electronics Laboratories, Stanford University, Stanford, CA, April 1977.
[26] H. J. Nussbaumer *Fast Fourier Transform and Convolution Algorithms*, Springer-Verlag, Berlin, 1981.
[27] G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1967.
[28] A. K. Jain, "Fast inversion of banded Toeplitz matrices by circular decompositions", *IEEE Acoust. Speech and Signal Proc.*, vol. ASSP-26, no. 2, pp. 121-126, April 1978.
[29] M. Morf, A. Vieira, D. T. Lee and T. Kailath, "Recursive multichannel maximum entropy method", *Proc. 1977 Joint Automatic Control Conf.*, San Francisco, CA, 1977.
[30] I. C. Gohberg, A. A. Semencul, "On the inversion of finite Toeplitz operators and their continuous analogs", Mat. Issled, vol. 2, pp. 201-233, 1972 (In Russian).
[31] I. C. Gohberg and I. A. Feldman, "Convolution Equations and Projection Methods for their solution", *Transl. of Math. Monographs*, vol. 41, Amer. Math. Soc., Providence, Rhode Island, 1974.
[32] T. Kailath, A. Vieira, M. Morf, "Inverses of Toeplitz Operators, Innovations and Orthogonal Polynomials", *SIAM Review*, vol. 20, 1978, pp. 106-119.
[33] O. D. Anderson, *Time Series Analysis and Forecasting: The Box-Jenkins Approach*, Butterworths, London.

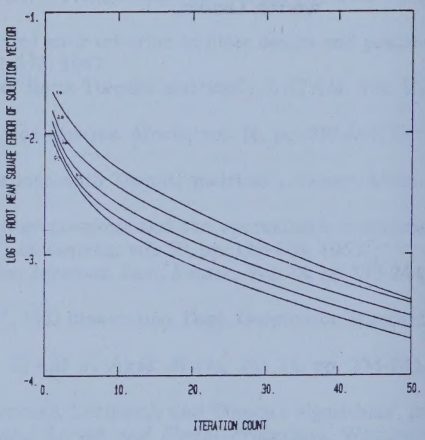




AR(2) PROCESS CONVERGENCE FOR ORDER 10, 20, 30, 40 AND 50 MATRICES
PHI1 = 0.10, PHI2 = 0.10



AR(2) PROCESS CONVERGENCE FOR ORDER 10, 20, 30, 40 AND 50 MATRICES
PHI1 = 0.45, PHI2 = 0.45



AR(2) PROCESS CONVERGENCE FOR ORDER 10, 20, 30, 40 AND 50 MATRICES
PHI1 = 0.00, PHI2 =-0.80

SYSTEM ORDER ESTIMATION: A SINGULAR VALUE DECOMPOSITION APPROACH

LEON H. SIBUL AND JOHN A. TAGUE
Applied Research Laboratory, The Pennsylvania State University,
Post Office Box 30, State College, Pennsylvania 16804

ABSTRACT

In the modeling approach to system identification, a system is often represented as an autoregressive process of appropriate order. The first step in such a modeling procedure is establishing a suitable model order. This paper presents an approach based on the singular value decomposition theorem.

INTRODUCTION

Given a finite sequence of data [y(n)] which represents the output of an unknown linear, stationary system of order p. We wish to model the unknown system as an autoregressive (AR) process whose input is white Gaussian noise [w(n)]. This work considers the problem of determining the system order p from [w(n)] and noisy measurements [y(n)].

APPROACHES TO THE PROBLEM

Several techniques have been developed to determine model order. One is based on covariance matrix rank determination[1] using singular value decomposition.[2] Cadzow (Reference [1]) and Konstantinides and Yao[3] discuss covariance matrix rank determination when the autocorrelation lags are estimated from noisy data.

ANOTHER APPROACH

The observations [y(n)] can be expressed in vector-matrix notation as:

$$\underline{y} = \underline{Y} \cdot \underline{b}$$

where

$$\underline{y} = [y(p'+1) \ \ldots \ y(N)]^T \ , \ \underline{b} = [a_1 \ \ldots \ a_{p'} \ 1]$$

and

$$\underline{Y} = \begin{bmatrix} y(p') & \ldots & -y(1) & w(p'+1) \\ \vdots & & \vdots & \\ y(N-1) & \ldots & y(N-p') & w(N) \end{bmatrix}$$

N is the number of samples, and p' is the estimated system order. If $p' > p$, then the columns of $\underline{Y}$ are no longer linearly independent, and $\underline{Y}$ is less than full rank. $\underline{Y}$ is of rank p+1 provided the measurements are known exactly. Therefore, system order can be estimated by performing a singular value decomposition on data matrix $\underline{Y}$ and examining its singular values. In addition, a least-squares estimate of the system parameters can be found by forming the pseudo-inverse $\underline{Y}^+ = (\underline{U}\underline{\Sigma}\underline{V}^T)^+ = \underline{V}\underline{\Sigma}^+\underline{U}^T$ and post-multiplying the result with data vector $\underline{y}$. $\underline{\Sigma}^+$ is found by replacing non-zero singular values by their reciprocals.

When [y(n)] is corrupted with measurement noise, $\underline{Y}$ is almost always of full rank because its columns become linearly independent no matter how large we choose p. But is is often possible to differentiate between large singular values which represent the "true" rank, and smaller singular values which represent measurement noise effects.[4] Therefore, model order and a least-squares parameter estimate are obtained directly from the observations without having to perform the intermediate step of estimating autocorrelation lags.

EXAMPLES AND NUMERICAL RESULTS

We present an example to compare the methods. Two hundred samples of white Gaussian noise were run through a filter with transfer function $H(z) = 1/A(z)$, with $A(z) = 1-3.4301z^{-1} + 4.8228z^{-2} -3.2599z^{-3} + 0.9031z^{-4}$. The results show that the data matrix has five to seven significant singular values, and the covariance matrix has four to six significant singular values. This suggests a model order of four to six should be selected, which matches well with the true order. Seven other autoregressive systems have been tested. So far, in all cases, both methods give comparable results.

This work has been supported by The Office of Naval Research under the Fundamental Research Initiatives Program.

REFERENCES

[1] J. A. Cadzow, "Spectral Estimation: An Overdetermined Rational Model Equation Approach," Proceedings of the IEEE, Vol. 70, No. 9, pp. 907-983, 1982.

[2] V. C. Klema and A. J. Laub, "The Singular Value Decomposition: Its Computation and Some Applications," IEEE Transactions on Automatic Control, Vol. AC-25, No. 2, 1980.

[3] K. Konstantinides and K. Yao, "Applications of Singular Value Decomposition to System Modeling in Signal Processing," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, San Diego, CA, March 1984.

[4] G. H. Golub and C. F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, 1983.

ON THE ROBUSTNESS OF AN ORDER-ESTIMATION TECHNIQUE

J.M. FLENTZ, Y.F. HUANG
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

Results are presented of two investigations of the robustness of an order-estimation technique for autoregressive moving-average models. After summarizing the technique and its extensions, the investigations are discussed. The first investigation is concerned with the application of the central-limit theorem. The second investigation is concerned with application of a series-expansion result that can be used in approximating the pdf used in the extended order-identification technique.

I. INTRODUCTION

As a continuation of development of a statistical order-determination algorithm presented in [1], investigations have been performed to determine its robustness. The automated technique is based upon a method developed by Suen and Liu [2]. The purpose of the continued study is to verify the feasibility of incorporation of the algorithm into an autonomous system. The system is being considered for use in the detection of frost/ice formation on the power rail of an electric mass-transit railway.

II. THE AUTOMATED SUEN-LIU METHOD

The system identification problem considered here is the determination of the structural parameters for the autoregressive moving-average (ARMA) equation

$$y_k = -\sum_{i=1}^{n} a_i y_{k-i} + \sum_{i=-L}^{n} b_i u_{k-i} + v_k \qquad (1)$$

$\{y_k\}$ is the observed output sequence of the system, $\{u_k\}$ is its input sequence, and $\{v_k\}$ is the observation-noise sequence. The system order and degree of noncausality are respectively denoted as n and L. The well-known batch least-squares technique of system identification is applied to a K-length input-output data sequence. A residual vector, the difference between the output and its estimate, is formed. The square of the norm of the residual vector can be written as an idempotent quadratic form in the observation noise vector. If the observation noise is iid $N(0,\sigma_v^2)$, the pdf of the norm-squared quantity is a variant of the χ^2 distribution. Parametric hypothesis-testing can be performed on the quantity for each (n,L) because the pdf is completely described. Details are in [1].

III. ROBUSTNESS OF THE TEST

The robustness of the automated Suen-Liu method has been studied in two ways. The first investigation involves application of the central-limit theorem. The second investigation is concerned with application of a series-expansion result [3] that can be used to directly approximate the pdf of e(n,L). Results of these studies are presented in this Section.

Central-Limit Approach

Application of the central limit theorem requires the norm-squared

quantity to be expressed as an idempotent quadratic form in the vector of output. $\{u_k\}$ and $\{v_k\}$ are each assumed to be zero-mean iid sequences. They are also mutually uncorrelated. Since y_k can be written entirely in terms of $\{u_k\}$ and $\{v_k\}$, the central-limit theorem guarantees asymptotic normality of $\{y_k\}$. A method is available ([4], p. 39) to approximate the pdf of a quadratic form of a Gaussian vector.

Asymptotic Approximation with Series of Functions

The above solution presents a motivation for the construction of the density function of the square of the norm of the residual vector when the observation noise is not Gaussian. The method to be considered now is based upon a result in [3] which is a series involving derivatives of the pdf of the observation noise. Since the series should be constructed from orthogonal basis functions, it is of interest to investigate the extent of applicability of this method. Based upon the assumption that $p_v(v)$, the pdf of the observation noise, is an element of the k-parameter exponential class ([5], p.72) of density functions, a theorem has been developed as a result of the study. Derivatives of $p_v(v)$ have the form

$$p_v^{(r)}(v) = \phi_r(v)p_v(v) \qquad (2)$$

The theorem states conditions for mutual orthogonality of the $\{\phi_i(v)\}$ on A, with respect to the weighting function $p_v(v)$. The conditions are:

(1) The pdf has moments of all (non-negative) orders, so that a set of orthogonal polynomials, $\{\psi_i(v)\}$, can be generated;

(2) $\phi_i(v) = a_i(\beta)\ \psi_i(v)$, where $\{a_i(\beta)\}$ is some set whose elements are restricted to constants and function of β.

Robustness of the technique with respect to observation-noise pdf's that satisfy the theorem can then be investigated via checking the closeness of the approximating pdf to the χ^2 pdf.

ACKNOWLEDGEMENT

This research has been supported in part by the Department of Transportation, Urban Mass Transit Administration, under Grant No. IN-06-0009.

REFERENCES

[1] Flentz, J.M, Y.F. Huang and W.J. Gajda, Jr., "The Statistical Properties of the Residual in System Identification", Proceedings of the Twenty-First Annual Allerton Conference Communication, Control, and Computing, pp. 712, 1983.

[2] Suen, L.C., and R. Liu, "A Normalized Residual Criterion for the Determination of Structure of Linear Systems", Journal of Policy Analysis and Information Systems, Volume 1, July 1977.

[3] Wallace, D.L., "Asymptotic Approximations to Distributions", Ann. Math. Statist., Vol. 29, pp. 635, 1958.

[4] Whittle, P., Hypothesis-Testing in Time Series Analysis, Uppsala: Almqvist and Wiksells, 1951.

[5] Bickel, P.J., and K.A. Doksum, Mathematical Statistics: Basic Ideas and Selected Topics, San Francisco: Holden Day, 1977.

A PERTURBATION APPROACH TO IMPROVING PISARENKO HARMONIC RETRIEVAL*

DANIEL R. FUHRMANN
Dept. of Electrical Engineering
Washington University
St. Louis, Missouri 63130

BEDE LIU
Dept. of Electrical Engineering
and Computer Science
Princeton University
Princeton, New Jersey 08540

ABSTRACT

One of the apparent drawbacks to eigenstructure-based (Pisarenko) spectrum estimation is the tendency of the resultant eigen-polynomial to have extraneous roots on the unit circle when the autocorrelation matrix R is overdetermined. This problem can be dealt with successfully by a complete eigenvector decomposition of R; however, this method can be computationally expensive. We show that similar results can be obtained by simply subtracting a small perturbation ε from the (0,0) element of R prior to finding the minimum eigenvector.

1. Introduction

This paper deals with the efficient implementation and approximation of the "improved Pisarenko" method of spectrum estimation.

Pisarenko [1] first introduced a method for extracting the parameters of a sinusoids-in-white-noise process using the exact autocorrelation matrix of that process. The algorithm as originally presented is not a very reliable method of parameter estimation. However, the attendant algebraic concepts have inspired a wide range of eigenstructure-based methods of spectrum estimation [2-3], source location [4], and adaptive filtering [5-7]. Most notable among these improvements are the sample (non-Toeplitz) covariance matrix, and matrices of order larger than the number of sinusoids in the signal model.

Kumaresan's "improved Pisarenko" (IP) method [2] is widely regarded as a superior method for the parameter estimation problem at hand. This technique requires the computation of the singular value decomposition (SVD) of a non-square Toeplitz data matrix, which is equivalent theoretically to the eigenvector decomposition of the sample covariance matrix $\hat{R}$. The method is computationally burdensome, as it requires $O(p^3)$ multiplications (p being the order of $\hat{R}$) and hence might be inappropriate for real-time implementation. This is particularly true for the adaptive PHR problem, where a sequence of sample covariance matrices is used to recursively compute the coefficients of an FIR adaptive notch filter [6,7].

The fundamental problem which the IP method attempts to solve is the resolution of ambiguities created by working with an $\hat{R}$ whose dimension is larger than the number of sinusoids in the original signal. The purpose of this paper is to demonstrate that these ambiguities can be resolved, to a good approximation, with a simple perturbation: subtract a small ε from the (0,0) element of $\hat{R}$ prior to eigenvector decomposition. The effect of this perturbation is to move the zeroes of the eigenvector polynomial inside the unit circle in such a way that the unwanted or extraneous zeroes are shifted further from the unit circle than the zeroes corresponding to the signals.

*This research was supported in part by the Air Force Office of Scientific Research under contract AFOSR-81-0186.

The paper will be organized as follows. In Section 2, Pisarenko's method of spectrum estimation and some of the signal/noise subspace concepts are reviewed. In Section 3, Kumaresan's IP method is explored in further detail. Section 4 describes the proposed perturbation method and shows that the zeroes of the perturbed eigenfilter lie inside the unit circle. Several examples are given to demonstrate the effectiveness of the method.

2. Review of Pisarenko's Method and Subspace Concepts

Pisarenko Harmonic Retrieval (PHR) [2] is a spectrum estimation technique designed for the specific problem of estimating the parameters of a signal x which can be realistically modeled by the following:

$$x(n) = \sum_{k=1}^{m} c_k e^{j(\omega_k n + \psi_k)} + w(n) \tag{2.1}$$

where w(n) is zero-mean white noise with variance σ^2. Usually the frequencies ω_k occur in conjugate pairs so that x is real, however, the analysis is simplified if the exponential form is used.

The true autocorrelation matrix of x is

$$R = \sum_{k=1}^{m} \rho_k \gamma_k \gamma_k^* + \sigma^2 I \tag{2.2}$$

where

$$\gamma_k = [\, 1 \;\; e^{j\omega_k} \;\; e^{j2\omega_k} \;\; \ldots \;\; e^{jp\omega_k} \,]^T . \tag{2.3}$$

R is (p+1)×(p+1) and $\rho_k = c_k^2$ (the power of each sinusoid). This decomposition exists for all Hermitian Toeplitz matrices as a result of Caratheodory's Theorem [8], and hence PHR can be implemented even when the signal model is not correct, although the results may not be meaningful.

Define the signal autocorrelation matrix S by

$$S = \sum_{k=1}^{m} \rho_k \gamma_k \gamma_k^* . \tag{2.4}$$

Since $m \leq p$, this matrix will be non-negative definite and singular, therefore its minimum eigenvalue is 0. It follows that the minimum eigenvalue of R is σ^2. The corresponding minimum eigenvector $a = [a(0) \ldots a(p)]^T$ will be orthogonal to each vector γ_k, that is,

$$\gamma_k^* a = 0, \qquad k = 1, \ldots m \tag{2.5}$$

Substituting the definition of γ_k from eqn. (2.2) we get

$$\sum_{i=0}^{p} a(i) e^{-j\omega_k i} = 0, \qquad k = 1, \ldots m \tag{2.6}$$

or equivalently, the polynomial $A(z^{-1}) = \sum_{i=0}^{p} a(i) z^{-i}$ has roots on the unit circle at angular frequencies ω_k. This polynomial is often referred to as the *eigenfilter*, since the FIR digital filter with tap coefficients a(i) would have notches at the frequencies ω_k.

Pisarenko's method consists of forming a sample Toeplitz autocorrelation matrix of order (m+1), finding its minimum eigenvector and performing a polynomial rooting to extract the frequencies. Unfortunately, this method is very sensitive to perturbations in the autocorrelation function, hence is really only applicable for long data records or high SNR. It also requires exact knowledge of the signal model order m.

The "improved Pisarenko", or IP, method, due to Kumaresan [3], is the result of two fundamental changes to the above algorithm: 1) the non-Toeplitz sample covariance matrix is used in place of the sample autocorrelation matrix, and 2) the order of the matrix R is chosen larger than m. For the remainder of this section we shall concentrate on the effect of the latter.

The basic philosophy behind using a larger p is straightforward: if we assume the sample autocorrelation function to be the true one corrupted by random variations due to the noise, then, the more autocorrelation lags that are available, the more accurate the parameter estimates. However, Pisarenko's method must be modified, since when $p > m$, the minimum eigenvalue of the true autocorrelation matrix is not simple, but has multiplicity (p+1-m). This is due to the fact that the rank of S is m regardless of its dimension. The eigenvectors of R corresponding to this eigenvalue σ^2 span a subspace of rank (p+1-m). Every vector in this subspace is orthogonal to each γ_k, hence every eigenfilter has m roots on the unit circle. The other (p-m) roots are not uniquely determined, which reflects the dimensionality of the minimum eigenspace.

The above observations have led to the following terminology: the γ_k are referred to as the *signal vectors* and the subspace given by *span* $\{\gamma_k\}$ is the *signal subspace*. This subspace has rank m. Its orthogonal complement is the space of minimum eigenvectors of R; this is referred to as the *noise subspace*. The noise subspace has rank (p+1-m).

The use of $p > m$ may improve the accuracy of the parameter estimates, or equivalently the notch locations of the eigenfilter, but it has also created an ambiguity: which of the vectors in the noise subspace is the appropriate one to use to best determine the m correct zero locations? The resolution of this ambiguity is central to the effective IP method. Kumaresan's solution, also due to Reddi [4], is to choose the *minimum-norm* eigenvector, that vector in the noise subspace which minimizes $\sum_{i=0}^{p} a^2(i)$ subject to $a(0) = 1$. It will be shown in a next section that this solution can be interpreted in number of interesting ways.

When the sample covariance matrix $\hat{R}$ is used in place of the true autocorrelation matrix, as must be done for finite data records, the (p+1-m) smallest eigenvalues are not identically σ^2 but rather will be clustered around that value. However, a good heuristic argument can be made to support the claim that the m largest eigenvectors of $\hat{R}$, i.e., those eigenvectors corresponding to the m largest eigenvalues, span the signal subspace of R to a very good approximation. The extent to which the eigenvectors are perturbed by the error introduced by the noise and the finite data record is dependent on the relative spacing of the eigenvalues (c.f. Wilkinson [5]). The eigenvalues corresponding to the signal subspace will be well-separated from the noise subspace, assuming either high SNR or a sufficient number of data points; hence this set of eigenvectors, or at least the invariant subspace which they span, is relatively stable. The noise subspace would be the orthogonal complement of these eigenvectors, and the IP eigenfilter is formed from that vector in the noise subspace which satisfies Kumaresan's criterion.

3. Properties of the IP Eigenfilter

In the previous section it was shown that, due to the dimensionality of the noise subspace, it is necessary to employ additional criteria in choosing a unique vector with which to form the eigenfilter. In what follows it will be shown that three rather natural criteria all lead to the same solution, which we refer to as the *IP eigenfilter*.

Consider the following three approaches to the problem of choosing a unique a from the noise subspace.

1) choose a to be minimum norm when a(0) = 1; i.e., choose a such that $\sum_{i=1}^{p} a_i^2$ is minimized subject to the constraint a(0) = 1,

2) choose a to be the orthogonal projection of $e_0 = [1\ 0\ 0\ \ldots\ ]^T$ onto the noise subspace N.

3) choose a such that the eigenfilter response $A(e^{-j\omega})$ is as close to all-pass (i.e., $A(e^{-j\omega})=1$) as possible, in a mean-square sense, subject to the constraints that $A(e^{-j\omega_k})=0$, for each k, and $\frac{1}{2\pi}\int_{-\pi}^{\pi}|A(e^{-j\omega})|^2 d\omega = 1$.

The first solution was offered by Kumaresan [2] and was motivated by the observation that the extraneous zeroes of $A(z^{-1})$ always lie inside the unit circle. The second criteria suggests a certain computational simplicity. The third approach states that the eigenfilter response should be as flat as possible outside of the notch regions. However, as the following theorem shows, the way in which one chooses to approach the problem is irrelevant.

THEOREM 1. The solution vectors a which satisfy the above three criteria are scalar multiples of one another; furthermore, this vector is unique to within a constant scale factor.

PROOF. We show that all three problem formulations are equivalent to a fourth, namely

$$\min |a-e_0|^2 \text{ s.t. } a\varepsilon N \text{ and } a^T = 1 \qquad (3.1)$$

1) $|a-e_0|^2 = a^Ta - 2a^Te_0 + e_0^Te_0 = 2(1-a^Te_0)$. Minimizing this quantity is equivalent to maximizing $a^Te_0 = a(0)$. Any vector with a(0) = 1 is a scalar multiple of a unit vector $\hat{a}$ such that $\hat{a}(0) = \frac{1}{(a^Ta)^{\frac{1}{2}}}$. The minimum norm a (with a(0) = 1) maximizes this quantity.

2) The fact that the unit vector a which maximizes a^Te_0 is the projection of e_0 onto N is a consequence of the projection theorem from linear algebra (c.f., Luenberger [9]).

3) We wish to minimize the mean-square difference between $(Ae^{-j\omega})$ and the response of an all-pass filter which is a constant (1).

$$\min \frac{1}{2\pi}\int_{-\pi}^{\pi}|1 - A(e^{-j\omega})|^2 d\omega \text{ s.t. } A(e^{-j\omega_k}) = 0\ ,\ k = 1\ \ldots\ m \qquad (3.2)$$

The constraint that $A(e^{-j\omega_k}) = 0$ is equivalent to $a\varepsilon N$. Define the vector $b = e_0 - a$; the z-transform of b is $1 - A(z)$. By Parseval's relation

$$|b|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |B(e^{-j\omega})|^2 d\omega \qquad (3.3)$$

and hence (3.1) and (3.2) are equivalent.

Uniqueness of a comes from solution (2), since the projection operator is simply a linear transformation acting on e_0. (End of proof).

Another way to describe the IP eigenvector is to say that it is the unit vector a in the noise subspace N with the largest (in magnitude) leading coefficient a(0).

One result of this theorem is a computationally simple way to compute the IP eigenvector: it is the projection of e_0 onto the orthogonal complement of the signal subspace. Assuming an orthogonal basis $(s_1, \ldots s_m)$ for this subspace, the eigenvector a could be computed by

$$a = (I - \sum_{k=1}^{m} s_k s_k^T) e_0. \qquad (3.4)$$

This requires only m(p+1) multiplications.

Vaccaro [7] has shown that, if the minimum eigenvector a is computed using the inverse power method on R with initial vector e_0, then the resulting solution will be the projection of e_0 onto the noise subspace. This result can be extended to any iterative mimimum-eigenvalue algorithm which computes a using linear combinations of terms of the form $R^n a$.

In [2] it was shown that the extraneous zeroes of $A(z^{-1})$ do indeed lie inside the unit circle in the z-plane. The third problem formulation here confirms and explains a second observation in [2], namely that the extraneous zeroes of $A(z^{-1})$ are roughly uniformly distributed in a circular pattern. Such would be the distribution of zeroes for an FIR notch filter defined by (3) above.

Although the IP method is a superior method for the estimation and filtering of harmonic signals in broadband noise, it is computationally expensive. The complete eigenvector decomposition of R requires $O(p^3)$ operations. This would make IP difficult to implement in a real-time situation, especially where a sequence of eigenfilters a(n) corresponding to a sequence of converging covariance matrices is desired. This computational burden provides the motivation for considering a simpler alternative to IP which will be presented in the remaining sections.

4. The Perturbation Approach

The perturbation method which is proposed is actually quite simple and was described earlier in Section 1: subtract a small amount ε from the (0,0) element of R or $\hat{R}$, then compute the unique minimum eigenvector of the perturbed matrix. We define $E_0 = e_0 e_0^T$ and hence can say that we seek the minimum eigenvector of $R - \varepsilon E_0$.

A heuristic argument to support this method is as follows: suppose one has a very good estimate of R such that the minimum eigenvalue σ^2 has multiplicity (p+1-m). To a first-order approximation the eigenvectors a_i of R are not affected by a small perturbation. The corresponding eigenvalues will be given by the Rayleigh Quotient $\hat{\lambda}_i = a_i^T (R - \varepsilon E_0) a_i$, assuming $a_i^T a_i = 1$.

For all eigenvectors a_i in the noise subspace N, $\lambda_i = \sigma^2$. Hence the unique vector which minimizes $\hat{\lambda}_i$ is the one which maximizes $a^T E_0 a$, or equivalently $a^2(0)$. This is of course the projection of e_0 onto N as was shown in the previous section.

Before proceeding we give an example to demonstrate the effectiveness of this technique: let R be a 32×32 symmetric Toeplitz matrix whose first row consists of 4 sinusoids (cosines) of normalized frequency 0.15, 0.16, 0.35, and 0.45, and amplitudes 1, 1, 1, and 0.5, respectively. Let $\varepsilon = 0.01$. The minimum eigenvector of $R - \varepsilon E_0$ was computed by an IMSL subroutine. The eigenpolynomial $A(e^{-j2\pi f})$ is shown in Fig. 1 for $f = 0$ to 0.5 . The notches are at the correct frequencies, and the extraneous zeroes are inside the unit circle such that the frequency response is as "all-pass" as possible.

One of the interesting things about this method is that if R is the exact autocorrelation matrix, and the lower (p+1-m) eigenvalues are all exactly σ^2, then ε can be arbitrarily small — and the smaller ε, the better the approximation to the desired eigenfilter. In practice, of course, the sample covariance or autocorrelation matrix has distinct eigenvalues, and ε will have be finite in order to shift the true eigenvector of a toward some vector which is close to e_0.

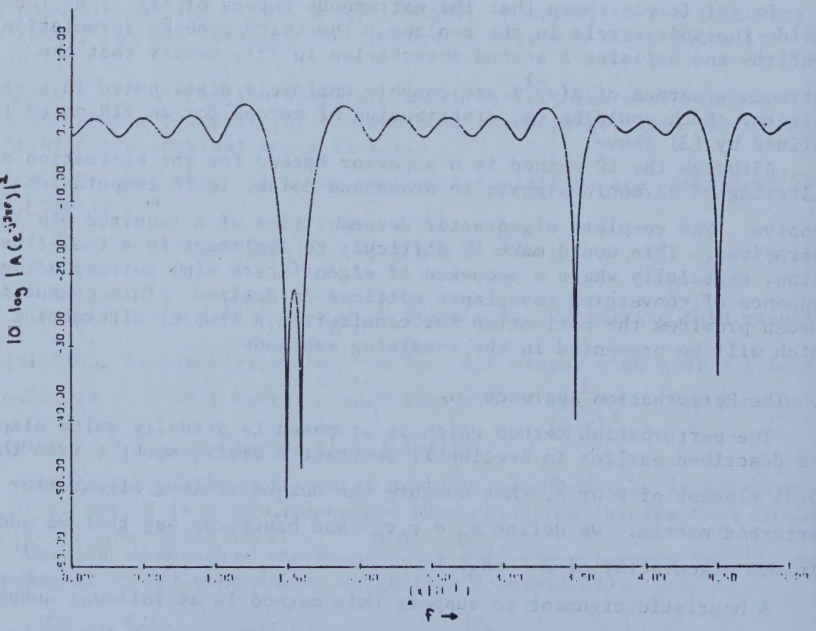


Fig. 1. Eigenfilter Response from Perturbation Technique with Exact Autocorrelation Matrix.

It will be comforting to know that the proposed perturbation will have a reliable effect on the eigenfilter $A(z^{-1})$. The following theorem is useful in that respect.

THEOREM 2. Let R be a positive definite Hermitian Toeplitz matrix. Let a be the minimum eigenvector of $R-\varepsilon E_0$, where $\varepsilon > 0$. Then the $\sum_{i=0}^{p} a(i)z^{-i}$ will have roots on or inside the unit circle in the z-plane.

PROOF. The proof follows from the minimum-delay property of finite sequences (c.f., [10], p. 80).

The minimum eigenvector of $R-\varepsilon E_0$ minimizes the Rayleigh Quotient $a^T(R-\varepsilon E_0)a$ over all unit vectors a. By the Caratheodory decomposition of R (eqn. 2.5) there exist m pairs (ρ_k,ω_k) such that

$$a^TRa = \sum_{k=1}^{m} \rho_k |\lambda_k^* a|^2 + \sigma^2 = \sum_{k=1}^{m} \rho_k |A(e^{j\omega_k})|^2 + \sigma^2 \qquad (4.1)$$

For a given realizable $H(\omega)$ there exist 2^p vectors in the set $\{a:|A(e^{-j\omega})|^2 = H^2(\omega)\}$ corresponding to whether or not each zero of $A(z^{-1})$ is inside or outside the unit circle. The *minimum-delay* a in this set has the property that the zeroes of $A(z^{-1})$ are on or inside the unit circle and a(0) is a maximum. Therefore any vector $\tilde{a}$ such that $\tilde{A}(z^{-1})$ has roots outside the unit circle cannot minimize the Rayleigh Quotient, since there would exist a corresponding equivalent minimum-delay a such that $\tilde{a}^TR\tilde{a} = a^TRa$ and $a^TE_0a > \tilde{a}^TE_0\tilde{a}$. (End of proof).

The above eigenfilter property will not always hold for $\hat{R}$, a sample covariance matrix (non-Toeplitz). This is due to the fact that the signal zeroes of R lie on the unit circle while those of $\hat{R}$ do not necessarily. However, as $\hat{R}$ approaches R, for sufficiently large ε we expect the same sort of behavior. Clearly as $\varepsilon \to \infty$ the minimum eigenvector approaches e_0 for which all the eigenfilter roots are at the origin.

The motivation behind this perturbation approach is the savings in computation which is achieved over the IP or other signal subspace methods. As described in [2], the IP method requires the computation of the singular value decomposition (SVD) of a p×(M-p) data matrix A. The singular vectors are the eigenvectors of the sample covariance matrix $\hat{R} = A^TA$. These decompositions require $O(p^3)$ multiplications in general. In contrast, there exist a number of good iterative algorithms to find the minimum eigenvector of a symmetric matrix, the current champion being the Rayleigh Quotient Iteration [12]. This algorithm exhibits cubic convergence, and requires the solution of a set of linear equations at each iteration. By the special structure of $\hat{R}$ this matrix inversion can be performed with $O(p^2)$ multiplications [13], and satisfactory convergence is usually achieved with just a few iterations.

5. Example

A number of simulations comparing the performance of the perturbation method with that of the IP method were performed. One example will be given here.

The following composite signal was used as the test data:

$$x(n) = \sum_{k=1}^{4} c_k \cos(2\pi f_k n + \psi_k) = w(n) \tag{5.1}$$

where $f_1 = 0.15$, $f_2 = 0.16$, $f_3 = 0.35$, and $f_4 = 0.45$. The power level ρ_k of each signal is $\frac{1}{2}c_k^2$; they were chosen to be $\rho_1 = 1$, $\rho_2 = 1$, $\rho_3 = 1$, and $\rho_4 = 0.1$. Initial phase ψ_k was 0 in all cases. $w(n)$ is simulated white Gaussian noise with variance σ^2. SNR is defined as $10 \log(1/\sigma^2)$; for the examples considered here, SNR was chosen to be 10 dB. The simulation program was written in Fortran and implemented with single precision arithmetic.

The 32×32 covariance matrix $\hat{R}$ was formed according to the estimate

$$\hat{R} = \sum_{i=1}^{200} x_i x_i^T \tag{5.2}$$

where

$$x_i = [x(i)\ x(i-1)\ \ldots\ x(i-p)]^T \tag{5.3}$$

This can be accomplished recursively with (p+1)M operations, p being 31 and M being 200 in this case. $\hat{R}$ is not properly normalized; however, this has no effect on the subsequent eigenvector calculations.

For the IP method, a complete eigenvector decomposition was performed via an IMSL Fortran subroutine. The eigenvectors $(z_1 \ldots z_8)$ corresponding to the 8 largest eigenvalues were chosen as the basis for the signal subspace. The IP eigenfilter was computed according to

$$a = (I - \sum_{i=1}^{8} z_i z_i^T) e_0 \tag{5.4}$$

This can be done with 8(p+1) operations. a was then normalized.

For the perturbation method, the smallest eigenvector of $(R-\varepsilon E_0)$ was computed. For this example, $\varepsilon/200$ was chosen to be 0.1.

The frequency response of the eigenfilter resulting from this simulation is shown in Figure 2. Fig. 2.a corresponds to the IP method, and Fig. 2.b corresponds to the perturbation method. The notches in both responses are clear. Note, however, that the filter response derived from the IP method is much closer to the "ideal" response of Fig. 1.

In further simulations it was seen that the IP method is effective at suppressing extraneous zeroes over a wide range of SNR. In contrast, the perturbation method exhibits a steady decrease in performance with decreasing SNR, as reflected in the larger fluctuations in the "passband" of the eigenfilter. At high SNR, the two methods have roughly the same performance.

6. Summary

A simple modification to Pisarenko Harmonic Retrieval has been proposed. In this method, the sample autocorrelation or covariance matrix R is replaced by $R-\varepsilon E_0$ prior to the minimum-eigenvector computation. The qualitative effect of this perturbation is to smooth the impulsive spectrum which results from PHR.

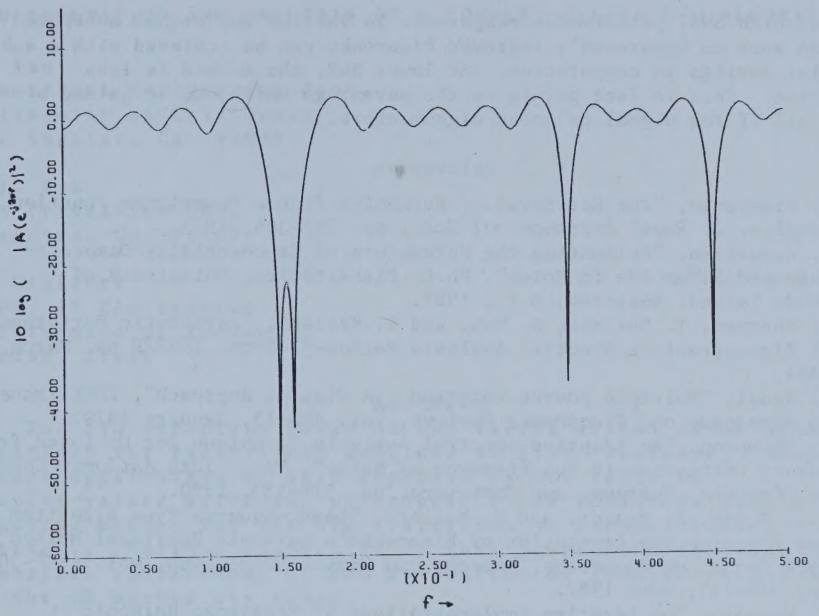


Fig. 2a. IP Method, SNR = 10 dB

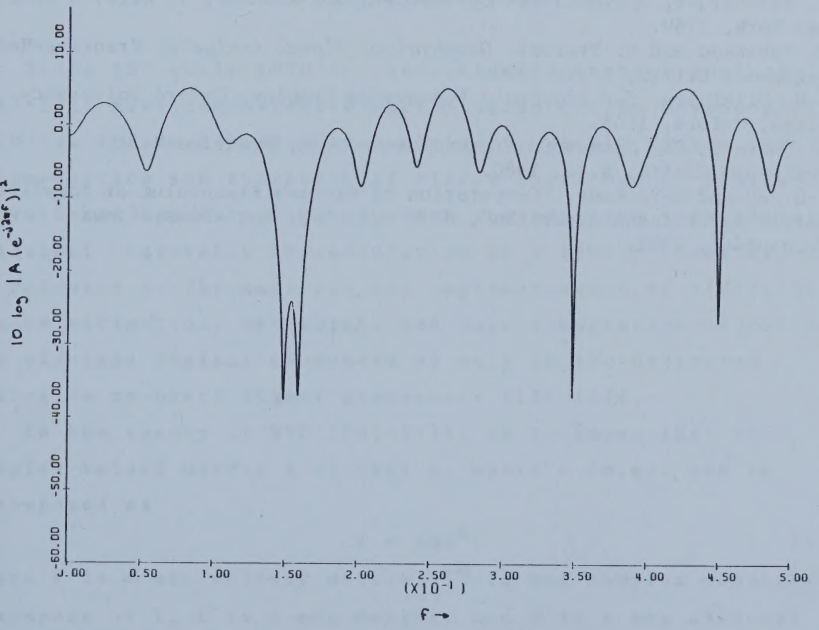


Fig. 2b. Perturbation Method, SNR = 10 dB

At high SNR, performance comparable to that of the signal subspace methods such as Kumaresan's Improved Pisarenko can be achieved with a substantial savings in computation. At lower SNR, the method is less effective. This in fact points to the advantage which can be gained by using all of the signal or noise eigenvectors.

References

1. V. Pisarenko, "The Retrieval of Harmonics from a Covariance Function", *Geophys. J. Royal Astronomical Soc.*, pp. 347-366, 1973.
2. R. Kumaresan, "Estimating the Parameters of Exponentially Damped or Undamped Sinusoids in Noise", Ph.D. Dissertation, University of Rhode Island, Kingston, R.I., 1982.
3. K. Sharman, T. Durrani, M. Wax, and T. Kailath, "Asymptotic Performance of Eigenstructure Spectral Analysis Methods", *Proc. ICASSP* 84, March 1984.
4. S. Reddi, "Multiple Source Location - A Digital Approach", *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-15, January 1979.
5. P. Thompson, "An Adaptive Spectral Analysis Technique for Unbiased Frequency Estimation in the Presence of Noise", *Proc. 13th Asilomar Conf. on Circuits, Systems, and Computers*, pp. 529-533, 1980.
6. V.U. Reddy, B. Egardt, and T. Kailath, "Least-Squares Type Algorithm for Adaptive Implementation of Pisarenko's Harmonic Retrieval Method", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, pp. 399-405, June 1982.
7. R. Vaccaro, "On Adaptive Implementations of Pisarenko Harmonic Retrieval", *ICASSP 84*, p. 6.1.1, March 1984.
8. U. Grenander and G. Szego, *Toeplitz Forms and Their Applications*, University of California Press, Berkeley, CA., 1958.
9. D. Luenberger, *Optimization by Vector Space Methods*, J. Wiley & Sons, New York, 1969.
10. E. Robinson and S. Treitel, *Geophysical Signal Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1980.
11. J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.
12. B. Parlett, *The Symmetric Eignevalue Problem*, Prentice-Hall, Englewood Cliffs, N.J., 1980.
13. Y-H. Hu and S-Y. Kung, "Computation of Minimum Eigenvalue of Toeplitz Matrix by Levinson Algorithm", *SPIE vol. 298, Real-Time Signal Processing*, 1981.

IMPLEMENTATION AND ANALYSIS OF A COMPLEX-VALUED SVD ALGORITHM

K. Yao
Electrical Engineering Dept.
University of California
Los Angeles, Ca. 90024

T.Y. Yan
Jet Propulsion Lab.
Pasadena, Ca. 91109

E. Biglieri
Dipt. di Elettronica
Politecnico di Torino
Torino, Italy

Abstract

The SVD technique based on the Jacobi method using plane rotations for real-valued matrices is first reviewed. Since a direct application of this approach is not valid for complex-valued matrices, we propose a new effective double rotation method for the SVD evaluation of complex-valued matrices. Various properties of this proposed method and numerical comparisons to the evaluation of complex-valued SVD by the QR method are given.

I. INTRODUCTION

Since the early 1970's, theoretical numerical analysists ([1]-[2]) have demonstrated that Singular Value Decomposition (SVD) is the most basic and numerically stable method for the decomposition and synthesis of linear operators on finite dimensional spaces. This approach includes and generalizes the classical eigenvalue representation of a linear operator. SVD is relevant in the analysis and implementation of efficient linear estimation, detection, and data compression algorithms for off-line digital computers as well as the dedicated real-time on-board signal processors ([3]-[5]).

In the theory of SVD ([6]-[7]), it is known that every mxn complex-valued matrix A of rank r, where r (m,n), can be decomposed as

$$A = USV^H, \tag{1.1}$$

where V is a nxn unitary matrix, V^H is the complex conjugate transpose of V, U is a mxn matrix, and S is a nxn diagonal matrix satisfying

$$V^HV = VV^H = I_n, \tag{1.2}$$

$$U^H U = I_n, \tag{1.3}$$

and

$$S = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0). \tag{1.4}$$

Then $\sigma_1, \ldots, \sigma_r$, are called the singular values of A.

Indeed, for every complex-valued mxn matrix A, there is a unitary matrix V satisfying (1.2) such that

$$(b_1, \ldots, b_n) = B = AV, \tag{1.5}$$

with orthogonal b_i's satisfying

$$b_i^H b_j = \sigma_i^2 \delta_{ij}. \tag{1.6}$$

But (1.6) can be expressed as

$$u_i = b_i / \sigma_i, \tag{1.7}$$

$$u_i^H u_j = \delta_{ij}, \tag{1.8}$$

$$\sigma_i \geq 0, \qquad i = 1, \ldots, n, \tag{1.9}$$

or

$$B = US, \tag{1.10}$$

with S given by (1.4) and U given by

$$U = [u_1, \ldots, u_n]. \tag{1.11}$$

From (1.7)-(1.11), we have U satisfying (1.3) but

$$UU^H = I_m. \tag{1.12}$$

Put (1.10) into (1.6),

$$US = AV. \tag{1.13}$$

Multiply (1.13) by V^H from the right results in the SVD of (1.1). Thus, given an arbitrary complex-valued A, the SVD problem is to find the SV and the U and V matrices.

II. REAL-VALUED SVD BY PLANE ROTATIONS

The method of plane rotations for solving eigenvalues was used by Jacobi [8] over a hundred years ago, but recently it has been extended to real-valued SVD by Nash [9]-[10]. The plane rotation approach represents probably the simplest method to describe the operations and the coding of a SVD algorithm. For some applications, it may indeed be the most useful implementational form of SVD. This approach is not necessarily optimum in terms of minimum number of arithematical operations,

minimum storage requirements, etc. In this section, we shall summarize the known results of real-valued SVD by plane rotations [9]-[10]. Various parts of this method are needed in the new complex-valued SVD method considered in the next section.

First, we consider the orthogonalization of two real-valued vectors by a single plane rotation. Consider two real-valued mx1 column vectors x and y. Let V be a 2x2 orthogonal matrix

$$V = \begin{bmatrix} \cos \emptyset & -\sin \emptyset \\ \sin \emptyset & \cos \emptyset \end{bmatrix} \tag{2.1}$$

such that the rotated vector X and Y are orthogonal. Let

$$[X,Y] = [x,y]V, \tag{2.2}$$

then

$$X = x \cos \emptyset + y \sin \emptyset, \tag{2.3}$$

$$Y = -x \sin \emptyset + y \cos \emptyset, \tag{2.4}$$

$$0 = X'Y = -x'x \cos \emptyset \sin \emptyset + x'y \cos^2 \emptyset$$

$$-y'x \sin^2 \emptyset + y'y \sin \emptyset \cos \emptyset \tag{2.5}$$

$$= (y'y-x'x)\sin \emptyset \cos \emptyset + x'y(\cos^2 \emptyset - \sin^2 \emptyset).$$

From (2.5), we have

$$(x'x-y'y)/x'y = 2(\cos^2 \emptyset - \sin^2 \emptyset)/\sin \emptyset \cos \emptyset . \tag{2.6}$$

Thus, the desired $\emptyset$ in (2.1) must satisfy (2.6). Since V is an orthogonal matrix, the following properties are valid:

1. $\|V\|^2 = \cos^2 \emptyset + \sin^2 \emptyset = 1,$ (2.7)

2. $X'X + Y'Y = x'x + y'y,$ (2.8)

3. For any real-valued mx1 column vector w,

$$(X,w)^2 + (Y,w)^2 = (x,w)^2 + (y,w)^2. \tag{2.9}$$

Consider

$$A_{k+1} = A_k V_k, \quad k = 1,2,..., \tag{2.10}$$

where V_k, k=1,2,..., are orthogonal matrices with A_1=A. Suppose after some finite N steps, B=A_{N+1} with component column vectors essentially mutually orthogonal. Then (1.5) can be

expressed as

$$B = AV, \tag{2.11}$$

$$V = V_1 V_2 \cdots V_N. \tag{2.12}$$

Since V_k, k=1,..., are orthogonal matrices, their products V is also an orthogonal matrix.

At each iteration in (2.9), we use a plane rotation orthogonal matrix

$$V(i,j) = \begin{array}{c} \\ i \\ \\ \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & i & & & & j \end{array} \\ \begin{bmatrix} 1 & & & & & & \\ & \cos\emptyset & & & -\sin\emptyset & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & \sin\emptyset & & & & \cos\emptyset & \\ & & & & & & 1 \end{bmatrix} \end{array} \tag{2.12}$$

We note this matrix consists of an identity matrix except the (i-j)th rows and columns are replaced by the orthogonal matrix of the form in (2.1). Thus by post-multiplying A_k by V(i,j), the i-th and j-th columns of A_{k+1} denoted by X and Y are made orthogonal. Specifically, let

$$A_{k+1} = [a_1,\ldots,a_{i-1},X,a_{i+1},\ldots,a_{j-1},Y,a_{j+1},\ldots,a_n] \tag{2.13}$$

$$A_k = [a_1,\ldots,a_{i-1},x,a_{i+1},\ldots,a_{j-1},y,a_{j+1},\ldots,a_n].$$

Then X and Y satisfy (2.3)-(2.5). While there are many ways for the rotation angle $\emptyset$ to satisfy (2.6), one approach ([9]) is to take

$$\cos\emptyset = [(v+q)/2v]^{.5}, \quad \sin\emptyset = p/v\cos\emptyset, \tag{2.14}$$

where

$$p = x'y, \quad q = x'x-y'y, \quad v = [4p^2+q^2]^{.5}. \tag{2.15}$$

At any iteration, the amount of non-orthogonalization is given by

$$z^{(k)} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (a_i'a_j)^2. \tag{2.16}$$

If $z^{(k)}=0$, then all columns of A_k are orthogonal and the orthogonalization process can terminate. Since only the i-th

and j-th columns are involved in the k-th iteration, the rest of the terms in the k-th and (k+1)st iterations are the same, then

$$z^{(k+1)} - (X'Y)^2 = z^{(k)} - (x'y)^2. \tag{2.17}$$

But by design $(X'Y)=0$, thus

$$z^{(k+1)} = z^{(k)} - (x'y)^2, \quad k = 1,... \tag{2.18}$$

This shows $z^{(k)}$ is a monotonically decreasing positive valued function. In one iteration, if the i-th and j-th column vectors are made orthogonal, in a subsequent iteration in making the i-th and m-th column vectors orthogonal, the original i-th and j-th column vectors are generally no longer orthogonal. However, in light of (2.18), $z^{(k)}$ is always decreasing and we are guaranteed of the convergence of the algorithm.

In a swquential computer, a sweep is defined by

$$V^{(0)} = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} V(i,j) , \tag{2.19}$$

where each V(i,j) is an orthogonal matrix of the form V_k in (2.12). We repeat over different sweeps until $z^{(k)}$ is sufficiently small. There are clearly $n(n-1)/2$ iterations per sweep. On the other hand, in a parallel computer, we can do

$$V(i,j),\ V(s,t),\ V(x,y),...,\quad i\neq j\neq s\neq t\neq x\neq y \ ... \ , \tag{2.20}$$

simultaneously. Thus, the speed of convergence should increase significantly.

III. NEW COMPLEX-VALUED SVD BY PLANE ROTATIONS

Now, suppose the mxn matrix A is complex-valued and we try to repeat the above technique for the new SVD. As we shall show now, it is not possible to make two complex-valued vectors orthogonal by the method of (2.1)-(2.5). Indeed, consider two complex-valued mx1 column vectors x and y. If we use V of (2.1), then X and Y are still defined by (2.3) and (2.4). However, orthogonality of complex-valued vectors are now defined by

$$0 = X^H Y = (y^H y - x^H x)\sin\phi\cos\phi + x^H y\cos^2\phi - y^H x\sin^2\phi, \tag{3.1}$$

where ^{H} denotes the complex conjugate transpose. Let

$$x^H y = \alpha + i\beta. \tag{3.2}$$

Then

$$y^H x = (x^H y)^H = \alpha - i\beta. \tag{3.3}$$

Thus, the orthogonality of (3.1) is equivalent to

$$0 = (y^H y - x^H x)\sin \phi \cos \phi + \alpha(\cos^2 \phi - \sin^2 \phi), \tag{3.4a}$$

$$0 = \beta(\cos^2 \phi + \sin^2 \phi) = \beta. \tag{3.4b}$$

From (3.4) we note that the orthogonality condition of X and Y is valid only if β, the imaginery part of $x^H y$, is zero. Of course, if x and y are both real as in Section 2, then the imaginery part of $x^H y$ is clearly zero. In general, for arbitrary complex-valued x and y, the imaginery part of $x^H y$ need not be zero, thus the transformation of the form of (2.1)-(2.2) do not yield orthogonal vectors.

We propose a new pre-processing scheme on the arbitrary input complex-valued vectors x and y such that the desired orthogonality condition of X and Y is satisfied. Let x be denoted by

$$x = [x_r, x_i], \tag{3.5}$$

where x_r and x_i are the real and imaginery parts of x. Let P be a new orthogonal 2x2 matrix

$$P = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \tag{3.6}$$

Let z be a complex-valued mx1 column vector with real parts z_r and imaginery part z_i defined by

$$z = [z_r, z_i] = [x_r, x_i]P. \tag{3.7}$$

Then

$$z_r = x_r \cos \theta + x_i \sin \theta , \tag{3.8a}$$

$$z_i = -x_r \sin \theta + x_i \cos \theta . \tag{3.8b}$$

Since P is an orthogonal matrix,

$$\|z\|^2 = \|x\|^2 . \tag{3.9}$$

Now, consider the original orthogonal matrix V of (2.1) operating on the pre-processed x of (3.7) and the arbitrary complex-valued input y

$$[Z, Y] = [z, y]V . \tag{3.10}$$

Then

$$Y = y \cos \phi - z \sin \phi, \tag{3.11}$$

$$Z = y \sin \phi - z \cos \phi. \tag{3.12}$$

Let the real parts of Y and Z be denoted by Y_r and Z_r and the imaginery parts of Y and Z be denoted by Y_i and Z_i. Then

$$Y_r = y_r \cos \phi - z_r \sin \phi, \tag{3.13}$$

$$Y_i = y_i \cos \phi - z_i \sin \phi, \tag{3.14}$$

$$Z_r = y_r \sin \phi + z_r \cos \phi, \tag{3.15}$$

$$Z_i = y_i \sin \phi + z_i \cos \phi. \tag{3.16}$$

As in (3.1) and (3.4b), in order for $Z^H Y=0$, we need

$$0 = -(z^H z - y^H y)\cos \phi \sin \phi + \{\mathrm{Re}[z^H y]\}(\cos^2 \phi - \sin^2 \phi), \tag{3.17}$$

$$0 = \mathrm{Im}\{[z^H y]\}. \tag{3.18}$$

Since

$$z^H y = (z_r - i z_i)'(y_r + i y_i)$$

$$= z_r' y_r + z_i' y_i + i(z_r' y_i - z_i' y_r), \tag{3.19}$$

then (3.18) becomes

$$z_r' y_i = z_i' y_r. \tag{3.20}$$

From (3.8a), (3.8b), and (3.20), (3.18) becomes

$$(x_r' \cos \theta + x_i' \sin \theta) y_i = (-x_r' \sin \theta + x_i' \cos \theta) y_r, \tag{3.21}$$

or

$$\sin \theta \, [x_i' y_i + x_r' y_r] = \cos \theta \, [x_i' y_r - x_r' y_i], \tag{3.22}$$

or

$$\theta = \tan^{-1} \{(x_i' y_r - x_r' y_i)/(x_i' y_i + x_r' y_r)\}. \tag{3.23}$$

Thus, by choosing θ given by (3.23), we satisfy (3.18).

In order to consider (3.17), we denote

$$q = z^H z - y^H y, \tag{3.24}$$

$$p = z_r' y_r + z_i' y_i, \tag{3.25}$$

$$v = (4p^2 + q^2)^{.5}. \tag{3.26}$$

Then (3.17) becomes

$$0 = -q \sin \phi \cos \phi + p(\cos^2 \phi - \sin^2 \phi). \tag{3.27}$$

There are various ways to solve for ϕ in (3.27). The method

used by Nash [9] for the real-valued case in Section 2 is still applicable here. Let

$$\cos \emptyset = [(v + q)/2v]^{.5}, \quad q \geq 0, \tag{3.28}$$

$$\sin \emptyset = p/(v \cos \emptyset), \quad q \geq 0, \tag{3.29}$$

$$\sin \emptyset = \operatorname{sgn}(p)\,[(v-q)/2v]^{.5}, \quad q < 0 \tag{3.30}$$

$$\cos \emptyset = p/v \sin \emptyset, \quad q < 0. \tag{3.31}$$

Direct substitutions of (3.28)-(3.31) into (3.27) shows the validity of the results.

IV. DISCUSSIONS

It is interesting to compare the new double rotation method presented in Section III with that of the known QR method [11] for complex-valued SVD. Just as there are various fine variations on the basic QR method, there are also various possible variations on the basic Jacobi method for eigenvalue and SV evaluations [7]. While the basic QR method is known to be an efficient numerical algorithm, this method is relatively complicated and coding for its implementation can be objectionable for many real-time signal processing and microcomputer applications. On the other hand, the basic Jacobi method yields a simple algorithm with easy to implement codes. Furthermore, the Jacobi method is ideally suited for parallel processing and VLSI implementations.

In Table I, we tabulate the actual cps time (seconds) used in an IBM PC computer (running Supersoft Fortran) for three versions of SVD with various dimensional complex-valued matrices. Results in the column under the QR method are based on that of Businger and Golub [11], while those under the double rotation method are based on the procedure presented in Section III with each sweep defined in the manner of (2.19). Results in the last column marked double rotation using largest inner product, are based on the double rotation method of Section III with each i-j index pair at each rotation given by

the largest magnitude of inner products of the distinct columns of the A_k matrix at each iteration. This variation of the Jacobi method tends to lead to faster mutual orthogonalization of the column vectors in A_{k+1} of (2.10). Of course, the checking of the largest magnitude of the inner products involves processing effort. All the original A matrices of different dimensions used in Table I are constructed from catenations of the basic 4x4 complex-valued rank 3 matrix given by Ex. 6.7 of Gregory and Karney [12].

As can be seen from these data, all three methods yield approximately similar processing time (within a factor of 2) for the relatively small values of n consider here. While some results on the analysis of the convergence of the different schemes exist [11], explicit comparison of different schemes appears to be possible only by direct computations. By direct compuations as well as from [11], it is known that for large dimensional matrices, the QR method is more advantageous than the Jacobi method from the convergence point of view. Of course, as stated earlier, from the implementation point of view, the Jacobi method has advantages over the QR method for various specific applications.

V. ACKNOWLEDGEMENTS

This work is partially supported by the Electronics Program of the Office of Naval Research and Cal Tech President's Fund.

VI. REFERENCES

1. G.H. Golub and W. Kahan, 'Calculating the Singular Values and Pseudo-Inverse of a Matrix,' SIAM J. Num. Anal. Ser. B 2, 1965, pp. 205-224.
2. G. Peters and J.H. Wilkinson, 'The Least Squares Problem and Pseudo-Inverses,' Computer Jour., Vol. 13, Aug. 1970, pp.309- 316.
3. H.C. Andrews and B.R. Hunt, 'Digital Image Restoration,' Prentice-Hall, 1977.
4. D.W. Tufts and R. Kumaresan, 'Singular Value Decomposition and Improved Frequency Estimation Using Linear Prediction,' IEEE Trans. on Acous., Speech, and Sig. Proc., Vol. ASSP-30, Aug. 1982, pp. 671-675.
5. V.C. Klema and A.J. Laub, 'The Singular Value Decomposition: Its Computation and Some Applications,' IEEE Trans. on Auto. Cont., Vol. AC-25, April 1980, pp.164-176.
6. C.L. Lawson and R.J. Hanson, 'Solving Least Squares

Problems,' Prentice-Hall, 1974.
7. G.H. Golub and C.F. Van Loan, 'Matrix Computations,' Johns Hopkins Press, 1983.
8. C.G.J. Jacobi, 'Uber ein Leichters Verfahren Die in der Theorie der Sacularstorungen Vorkommendern Gleichungen Numerisch Aufzulosen,' Crelle's Jour., Vol. 30, 1846, pp.51-94.
9. J.C. Nash, 'A One-Sided Transformation Method for the Singular Value Decomposition and Algebraic Eigenproblem,' Comp. Jour., Vol. 18, 1975, pp. 74-76.
10. J.C. Nash, 'Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation,' J. Wiley, 1979.
11. P.A. Businger and G.H. Golub, 'Algorithm 358: Singular Value Decomposition of a Complec Matrix,' Commun. Assoc. Comp. Mach., Vol. 12, 1969, pp. 564-565.
12. R.T. Gregory and D.L. Karney, 'A Collection of Matrices for Testing Computational Algorithms,' J. Wiley, 1969.

Matrix Dimen.	QR Method	Double Rotation Method	Double Rotation Method Using Largest Inner Pr.
16 x 8	11.53	20.10	14.17
12 x 8	8.95	15.22	10.66
16 x 4	3.35	5.00	2.69
12 x 4	2.63	3.74	2.40
8 x 4	1.98	2.80	1.37
4 x 4	1.26	1.38	0.72

Table I. Processing Time (Seconds) for Complex-Valued SVD

LOWER BOUNDS ON THE MONOTONE NETWORK COMPLEXITY OF THRESHOLD FUNCTIONS

PAUL. E. DUNNE

Department Of Computer Science

University Of Warwick

Coventry CV4 7AL

Great Britain

ABSTRACT

Let $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ be a set of n boolean variables. The k-th *threshold function* (T_k^n), is the monotone boolean function which is defined to be 1 if at least k inputs are 1, and to be 0 otherwise. In this paper we prove:

$\forall\ 3 \leq k \leq \lceil n/2 \rceil$:

$$\mathbf{C^m}(\ T_k^n\ ) \geq \max\{\ 2n+3k\ ,\ 2.5n+1.5k\ \} - \mathbf{C}$$

where $\mathbf{C}$ is a constant and $\mathbf{C^m}(f)$ denotes the monotone network complexity of f.

This result implies a lower bound of $3.5n$ on the Majority function ($T^n_{\lceil n/2 \rceil}$ or MAJ_n), improving the previous lower bound of $3n$ due to Bloniarz [1]. The proof approach employed is a generalisation of the standard inductive gate elimination argument. We also utilise the result of Dunne [2], but not the methods. □

1. Definitions

Definition 1

A *monotone boolean network* $\mathbf{S}$ is a directed acyclic graph containing two disjoint labelled sets of nodes: $\mathbf{X}$ (the *inputs* of $\mathbf{S}$) is the set of nodes having in-degree 0. These are labelled with members of the set $\{x_1, \ldots, x_n\}$. $\mathbf{G}$ is the set of nodes having in-degree 2, the *gates* of $\mathbf{S}$. Gates are labelled by $\wedge$ (Boolean conjunction) or $\vee$ (Boolean disjunction). $\mathbf{S}$ contains a unique node $\mathbf{t}$ having out-degree 0, called the *output* of $\mathbf{S}$. The out-degree of an arbitrary node u is called the *fan-out* of u.

Each node g computes some monotone boolean function $RES(g)$, in a natural manner. □.

Definition 2

Define $\Delta(T_k^n)$ to be $n/2 - k$. Δ represents the *drift* of T_k^n from MAJ_n and may be negative and non-integral.

□

This research was supported by the Science & Engineering Research Council of Great Britain.

Definition 3

Let **S** be a monotone network computing T_k^n. Let π be a partial assignment such that $\mathbf{S} \xrightarrow{\pi} \mathbf{S}'$, i.e $\mathbf{S}|_\pi = \mathbf{S}'$, where $\mathbf{S}'$ computes $T_k^{n,-r}$. The *descriptor* of π, $\delta(\pi)$, is a triple (r,s,t) where:

$r = |\{$ *Inputs of* **S** *set by* $\pi\}|$

$s = \Delta(T_k^n) - \Delta(T_k^{n,-r})$

$t \leq |\{$ *Gates deleted from* **S** *by applying* $\pi\}|$ □

Definition 4

An $\alpha\beta$-reduction for T_k^n, is a set of q descriptor pairs, $\{<\alpha_i,\beta_i>\}$ such that:

For any **S** computing T_k^n, $\exists <\alpha_i,\beta_i>$ and partial assignments π, π' applicable to **S** for which:

$$<\delta(\pi),\delta(\pi')> \in \{<\alpha_i,\beta_i>\} \tag{1}$$

$$\forall <\alpha_i,\beta_i> \ (s_i+s_i') = 0 \tag{2}$$

□

2. Preliminary Results

Lemma 1

Let **S** compute T_k^n and let $\{<\alpha_i,\beta_i>\}$ be an $\alpha\beta$-reduction for **S**. Let $\Delta(T_k^n) = s$. If there is a function $\varphi(n,s) \rightarrow \mathbf{Q}^+$ such that:

$$\varphi(n,s) \leq \max\begin{cases} \varphi(n-r_i,s+s_i) + t_i \\ \\ \varphi(n-r_i',s+s_i') + t_i' \end{cases} \tag{*}$$

$\forall <\alpha_i,\beta_i> \equiv <(r_i,s_i,t_i),(r_i',s_i',t_i')>$

then:

$$\mathbf{C^m}(T_k^n) \geq \varphi(n,s)$$

Proof

By Induction on n

Base

Obvious for suitable base values of $\varphi(n,s)$

Step

We assume $\forall n' < n, \forall k'$ that:

$$\mathbf{C^m}(T_{k'}^{n'}) \geq \varphi(n',s') \ ; \ \Delta(T_{k'}^{n'}) = s'$$

and show that this implies the result.

Thus let **S** be a monotone network computing T_k^n. As $\{<\alpha_i,\beta_i>\}$ is an $\alpha\beta$-reduction, there exist partial assignments π, π', applicable to **S**, such that:

$$< \delta(\pi), \delta(\pi') > \; = \; < \alpha_i, \beta_i >$$

for some $1 \leq i \leq q$.

Thus:

$$\mathbf{C^m}(T_k^n) \geq \max \begin{cases} \mathbf{C^m}(T^{n-r_i}_{(\frac{n-r_i}{2})+s+s_i}) + t_i \\ \\ \mathbf{C^m}(T^{n-r_i'}_{(\frac{n-r_i}{2})+s+s_i'}) + t_i' \end{cases}$$

By the inductive hypothesis:

$$\mathbf{C^m}(T_k^n) \geq \max \begin{cases} \varphi(n-r_i, s+s_i) + t_i \\ \\ \varphi(n-r_i', s+s_i') + t_i' \end{cases}$$

But:

$$\varphi(n,s) \leq \max \begin{cases} \varphi(n-r_i, s+s_i) + t_i \\ \\ \varphi(n-r_i', s+s_i') + t_i' \end{cases}$$

Hence: $\mathbf{C^m}(T_k^n) \geq \varphi(n,s)$ □

Lemma(1) yields a recurrence expression for the monotone network complexity of T_k^n. We do not attempt to find a general solution to this, but illustrate that a particular $\varphi(n,s)$ is given by a specified $\alpha\beta$-reduction.

Lemma 2

If:

$$\begin{aligned} \mathbf{AB} = \{ & < (1, 1/2, 4), (1, -1/2, 3) >, && (1) \\ & < (1, 1/2, 5), (1, -1/2, 2) > && (2) \\ & < (2, 1, 8), (2, -1, 6) > && (3) \\ & < (1, 1/2, 3), (1, -1/2, 4) > && (4) \\ & < (1, 1/2, 2), (1, -1/2, 5) > && (5) \\ & < (2, 1, 6), (2, -1, 8) > \} && (6) \end{aligned}$$

is an $\alpha\beta$-reduction for every **S** computing $T^n_{n/2-s}$ then:

$$\varphi(n,s) = \begin{cases} 3.5n - |s| - \mathbf{C} & 0 \leq |s| \leq 3/2 \\ \\ 3.5n - 3|s| + 3 - \mathbf{C} & |s| \geq 3/2 \end{cases}$$

satisfies:

$$\varphi(n,s) \leq \max \begin{cases} \varphi(n-r_i, s+s_i) + t_i \\ \\ \varphi(n-r_i', s+s_i') + t_i' \end{cases}$$

$\forall\ <\alpha_i,\beta_i> \equiv <(r_i,s_i,t_i),(r_i',s_i',t_i')> \in \mathbf{AB}$

Proof

By inspection. □

It may be noted that in some $<\alpha_i,\beta_i>$:

$$\varphi(n,s) \geq \min\begin{cases}\varphi(n-r_i,s+s_i)+t_i\\ \varphi(n-r_i',s+s_i')+t_i'\end{cases}$$

e.g $\varphi(n,1/2) > \varphi(n-2,-1/2)+6$

This imposes a strategy in inductively eliminating gates from **S** computing $T^n_{\lfloor n/2\rfloor-s}$, in that for those $<\alpha_i,\beta_i>$ having this property the step which reduces to:

$$\varphi(n,s) \leq \max\begin{cases}\varphi(n-r_i,s+s_i)+t_i\\ \varphi(n-r_i',s+s_i')+t_i'\end{cases}$$

must be applied.

Lemma 3

If **S** computes T^n_k and $k \notin \{1,n\}$, then every input x_i entering a gate g at a maximal distance from the output of **S** has fanout≥2.

Proof

g must have inputs x_i, x_j. If the fanout of x_i equals 1, then there is a partial assignment $x_j = \mathbf{c}$ $\mathbf{c} \in \{0,1\}$, such that $\mathbf{S}^{|x_j=\mathbf{c}}$ does not depend on x_i. But $\mathbf{S}^{|x_j=\mathbf{c}} = T^{n-1}_{k'}$ where $k' \in \{k,k-1\}$, and this function still depends on x_i if $k \notin \{1,n\}$. Contradiction.

□

3. Main Result

Theorem 1

Let **S** be any optimal network computing T^n_k for $1<k<n$. Then **AB** is an $\alpha\beta$-reduction for **S**.

Proof

Let g_2 be a gate of **S** at a maximal distance from the output. The inputs of g_2 must be distinct inputs x_i, x_j of **S**. We proceed by case analysis on the environment of x_i and show that in each case some $<\alpha_i,\beta_i> \in \mathbf{AB}$ is applicable.

Case 1 fanout(x_i)=1

If $1 < k \leq n-1$, then from Lemma(3) this case cannot occur.

Case 2 fanout(x_i) $\geq$3

There must exist some constant **c** $\in \{0,1\}$ such that setting x_i = **c** eliminates at least 5 gates, as two of the gates entered must have the same operation. But, since the fanout of $x_i \geq 3$, setting $x_i = \neg$**c** must eliminate at least 3 gates. Thus:

$$<\delta(x_i=1),\delta(x_i=0)> \in \{<\alpha_2,\beta_2>,<\alpha_5,\beta_5>\}$$

Case 3 fanout(x_i)=2

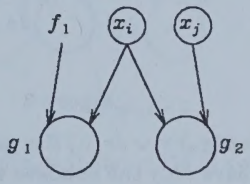


Figure 1

3.1) op(g_1) = op(g_2), wlog op(g_1)=$\vee$

3.1.1) fanout(g_1)$\geq$2 or fanout(g_2)$\geq$2

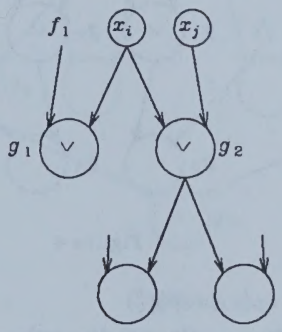


Figure 2

$$<\delta(x_i=1),\delta(x_i=0)> = <\alpha_2,\beta_2>$$

3.1.2) successor(g_1)=successor(g_2)

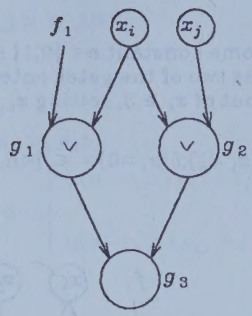


Figure 3

In both possible cases ($op(g_3) = \vee$ *or* $\wedge$) **S** is not optimal.

(3.1.1) and (3.1.2) leave only the subcase where g_1 has a unique successor g_3, and g_2 has a unique successor g_4 with $g_4 \neq g_3$.

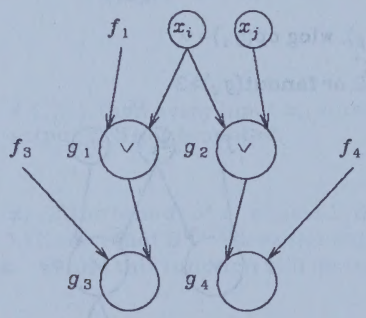


Figure 4

3.1.3) $op(g_3)=op(g_1)$ or $op(g_4)=op(g_2)$

Then: $<\delta(x_i=1),\delta(x_i=0)> = <\alpha_2,\beta_2>$

Thus $op(g_3)=op(g_4)=\wedge$

Now, by the choice of g_2, the other function, f_4 say, input to g_4 is either some input x_k of **S** or the output of a gate at a maximal distance from the output of **S**.

3.1.4) $f_4 = x_k$

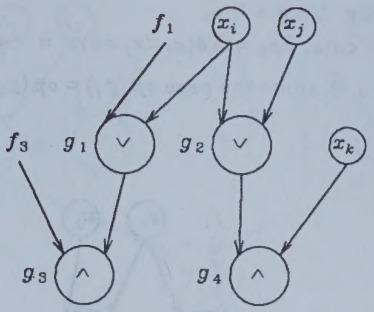


Figure 5

$$<\delta(x_i=1),\delta(x_k=0)> = <\alpha_1,\beta_1>$$

3.1.5)

f_4 is the output of a gate h_1 with inputs x_k , x_l

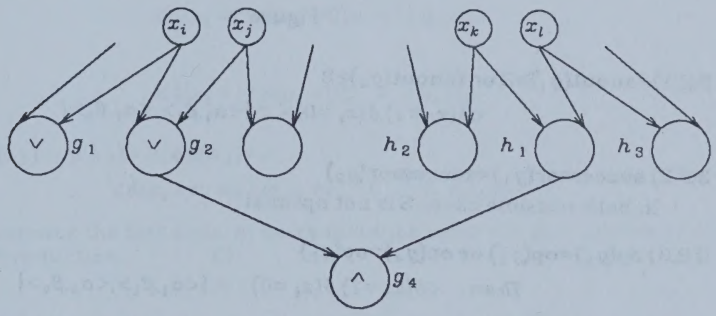


Figure 6

3.1.5.1) op(h_1)=∧

$$<\delta(x_k=1),\delta(x_k=0)> = <\alpha_5,\beta_5>$$

3.1.5.2) op(h_2)=∧ or op(h_3)=∧

$$\left.\begin{array}{l} <\delta(x_i=1),\delta(x_k=0)> \\ or \\ <\delta(x_i=1),\delta(x_l=0)> \end{array}\right\} = <\alpha_1,\beta_1>$$

3.1.5.3) $\mathbf{op}(h_q)=\vee \quad \forall q \quad 1 \le q \le 3$

$$<\delta(x_i=x_k=1),\delta(x_i=x_j=0)> \;=\; <\alpha_3,\beta_3>$$

This exhausts Case(3.1), since the case $op(g_1) = op(g_2) = \wedge$ follows by a dual argument.

3.2) $\mathbf{op}(g_1)\neq op(g_2)$

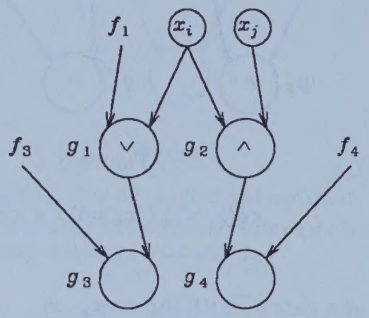


Figure 7

3.2.1) $\mathbf{fanout}(g_1)\geq 2$ **or** $\mathbf{fanout}(g_2)\geq 2$

$$<\delta(x_i=1),\delta(x_i=0)> \in \{<\alpha_1,\beta_1>,<\alpha_4,\beta_4>\}$$

3.2.2) $\mathbf{successor}(g_1)=\mathbf{successor}(g_2)$

In both possible cases **S** is not optimal

3.2.3) $\mathbf{op}(g_3)=\mathbf{op}(g_1)$ **or** $\mathbf{op}(g_4)=\mathbf{op}(g_2)$

$$\textit{Then}: \quad <\delta(x_i=1),\delta(x_i=0)> \in \{<\alpha_1,\beta_1>,<\alpha_4,\beta_4>\}$$

Thus as in Case(3.1), $f_4 = x_k$ or f_4 is the output of a gate at maximal distance from the output of **S**.

3.2.4) $f_4 = x_k$

Similar to subcase(3.1.4) above

3.2.5)

f_4 is the output of gate h_1 with inputs x_k , x_l

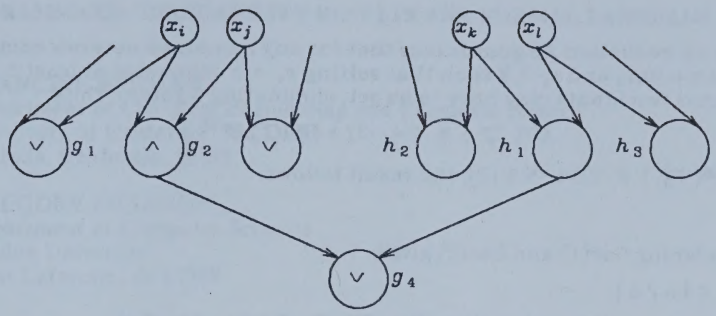


Figure 8

3.2.5.1) op(h_1)=$\vee$

$$<\delta(x_k=1),\delta(x_k=0)> \;=\; <\alpha_2,\beta_2>$$

3.2.5.2) op(h_2)=$\wedge$ or op(h_3)=$\wedge$

$$<\delta(x_i=1),\delta(x_k=0)> \;=\; <\alpha_4,\beta_4>$$

or:

$$<\delta(x_i=1),\delta(x_l=0)> \;=\; <\alpha_4,\beta_4>$$

3.2.5.3) op(h_2)=op(h_3)=$\vee$, op(h_1)=$\wedge$

$$<\delta(x_i=x_j=1),\delta(x_i=x_k=0)> \;=\; <\alpha_3,\beta_3>$$

This removes the last case. In every instance some $<\alpha_i,\beta_i>$ applies and thus **AB** is an $\alpha\beta$-reduction. □

4. Consequences Of Theorem(1)

Corollary 1

$\forall\, k \;\; 3 \le k \;\lceil n/2 \rceil$

$$\mathbf{C^m}(T_k^n) \;\ge\; 2n + 3k - \mathbf{C} \qquad \mathbf{C} \in \mathbf{Q^+}$$

Proof

Substitute $k = \lceil n/2 \rceil - s$ in the expression of Lemma(2). □

Corollary 2)

$\forall\, k \;\; 3 \le k \le \lceil n/2 \rceil$

$$\mathbf{C^m}(\, T_k^n \,) \;\ge\; 2.5n + 1.5k - \mathbf{C}$$

Proof

The $\alpha\beta$-reduction **AB** guarantees that for any monotone network computing T_k^n, $\exists\ \mathbf{c} \in \{0,1\}$ and $x_i \in \mathbf{X}$ such that setting $x_i = \mathbf{c}$ eliminates at least 4 gates. (Sometimes two inputs may have to be set, eliminating 8 gates). Thus:

$$\mathbf{C^m}(\ T_k^n\ ) \geq 4(k-3) + \mathbf{C^m}(\ T_3^{n-k+3}\ )$$

Since: $\mathbf{C^m}(\ T_3^n\ ) \geq 2.5n - 5.5$ [2], the result follows.

□

Combining Cor(1) and Cor(2) gives:

$\forall\ 3 \leq k \leq \lceil n/2 \rceil$

$$\mathbf{C^m}(\ T_k^n\ ) \geq \max\{2n+3k-\mathbf{C}_0\ ,\ 2.5n+1.5k-\mathbf{C}_1\ \}$$

□

Acknowledgement

I would like to thank Prof. M.S. Paterson for several valuable ideas concerning these results.

5. References

[1] P.Bloniarz: Ph.D Thesis; M.I.T, 1979

[2] P.E. Dunne: A 2.5n Lower Bound On The Monotone Network Complexity Of T_3^n; Theory Of Computation Report No.62, Dept Of Computer Science, Univ. Of Warwick; March 1984; (To appear)

EDGE GRAMMARS: DECIDABILITY RESULTS AND FORMAL LANGUAGE ISSUES

FRANCINE BERMAN
Department of Electrical Engineering and Computer Science
University of California, San Diego
La Jolla, California 92093

GREGORY SHANNON
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

ABSTRACT

Edge grammars are a formal mechanism for describing graphs and graph families. In this paper, we investigate decidability problems for edge grammars and in particular, the membership problem: "Given a graph H and an edge grammar Γ, is H isomorphic to a member of G(G), the graph family generated by Γ?" We show that while this problem is undecidable in general, it is decidable for a class of edge grammars whose graph families are commonly used in parallel computation. We also relate in this paper edge grammar types and formal grammars and show that the edge grammar hierarchy complements the Chomsky hierarchy in a sawtoothed fashion.

MOTIVATION

Graphs and graph families are a fundamental tool in almost every area of computer science. They are used to represent algorithms, architectures, data structures, data flow, automata, etc.

Many applications use graphs and graph families as an abstraction and require a formal means of defining graphs, in the same way that context-free grammars are used to defined Type 2 languages in the construction of compilers. In [1], we introduced *edge grammars* as a formalism in which to define and manipulate graphs and graph families. Essentially, edge grammars are systems which define graph families by generating pairs of strings which represent edges in the individual graphs. These pairs are generated by derivations using the productions of the edge grammar in much the same way strings in conventional formal languages are generated. Edge grammars were developed to fill a gap left by other graph generating systems: In particular, we were interested in representing graph families commonly used as interconnection networks in parallel computation, (e.g. shuffle-exchange graphs, cube-connected cycles, meshes, trees, etc.) many of which could not be represented by existing graph grammars and graph generating systems [3], [4]. To this end, we explored the relationship of edge grammars to parallel computation in the introductory paper.

In this paper, we provide a solid context for the study of edge grammars by relating edge grammars to conventional formal language theory and to the existing work on graph grammars. We also explore decidability issues for edge grammars. In presenting the decidability results, we come full circle and resolve questions important in the motivating applications.

SECTION 1: DEFINITIONS

In this section, we give a brief review of edge grammars and an example of a graph family generated by one. For a more thorough description of edge grammars, see [1].

Definition: A (Type 0) **edge grammar** is a 4-tuple $<N,T,G,P>$ where N is a set of nonterminal symbols, T is a set of terminal pairs $\{(v,w)\}$ (v and w are strings over a finite alphabet), G in N is the start symbol and P is a set of productions. Productions in P have the form $x \rightarrow y$ where x and y are strings in $(NUT)^*$ and x is not empty. Note that we interpret the concatenation of two pairs (a,b)(c,d) to be the pair (ac,bd).

Essentially, a Type 0 edge grammar is exactly like a Type 0 (Chomsky) grammar except that instead of single terminal symbols, edge grammars have terminal pairs.

Definition: An edge grammar is **Type 3** (right linear) if the productions have the form $A \rightarrow B$, $A \rightarrow (a,b)B$ or $A \rightarrow (a,b)$ where A and B are nonterminals and (a,b) is a terminal pair; **Type 2** if the productions have the form $A \rightarrow xy$ or $A \rightarrow x$ where A is a nonterminal and x and y are either nonterminals or terminal pairs; and **Type 1** if for every production in P, the number of terminals and nonterminals on the left-hand side of the production does not exceed the number of terminals and nonterminals on the right-hand side.

Definition: Let Γ be an edge grammar. The nth graph generated by Γ, $\mathbf{G_n}$, is the undirected graph with vertices

$$\mathbf{V_n} = \{v \mid \text{For some } w, (G \rightarrow {}^*(v,w) \text{ or } G \rightarrow {}^*(w,v)) \text{ and } |v|=|w|=n\}$$

and edges

$$\mathbf{E_n} = \{(v,w) \mid G \rightarrow {}^*(v,w),\ v \neq w \text{ and } |v|=|w|=n\}.$$

In other words, the nth graph G_n is the graph all of whose edges have length n labels.

Definition: Let Γ be an edge grammar. The graph family generated by Γ, $\mathbf{G(\Gamma)}$ is the set $\{G_n\}_{n>0}$ where G_n is the nth graph generated by Γ.

Edge grammars generate pairs of labels. If those labels have the same length, say length n, then we consider them to be an edge in graph G_n. In this way, the edges generated by an edge grammar can be naturally partitioned into a family of graphs $\{G_n\}$. Note that the index n in G_n refers to the length of the labels in G_n and not necessarily the number of nodes in G_n.

To illustrate how edge grammars work, consider the family of shuffle-exchange graphs. A shuffle-exchange graph SE_n consists of 2^n vertices. Each vertex is labeled by a binary n bit string. Adjacent vertices have labels which are either left or right (circular) shifts of one another (the shuffle edges) such as (1000, 0001) and (1000, 0100), or have labels in which the last bit is complemented (the exchange edges) such as (1000, 1001). Shuffle-exchange graphs are particularly effective as parallel architectures for algorithms for sorting, fast fourier transform, performing permutations, etc. [8], [9]. Shuffle-exchange graphs SE_1, SE_2 and SE_3 are shown in Figure 1.

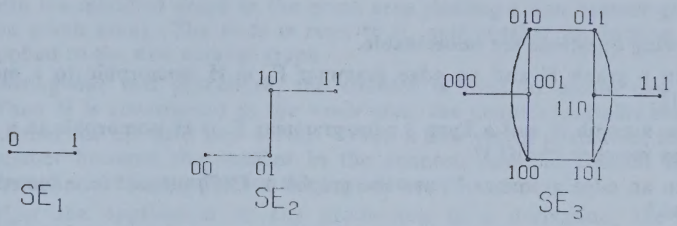


Figure 1

Shuffle-exchange graphs SE_1, SE_2 and SE_3.

The following is a (Type 3) edge grammar for generating the family of shuffle-exchange graphs $\{SE_n\}$.

$G \rightarrow E$	$G \rightarrow S_1$	$G \rightarrow S_0$
$E \rightarrow (0,0)E$	$S_1 \rightarrow (1,\lambda)N_1$	$S_0 \rightarrow (0,\lambda)N_0$
$E \rightarrow (1,1)E$	$N_1 \rightarrow (0,0)N_1$	$N_0 \rightarrow (0,0)N_0$
$E \rightarrow (0,1)$	$N_1 \rightarrow (1,1)N_1$	$N_0 \rightarrow (1,1)N_0$
	$N_1 \rightarrow (\lambda,1)$	$N_0 \rightarrow (\lambda,0)$

Edges in $\{SE_n\}$ are generated by applying a sequence of productions as in conventional formal language theory. For example, the following derivation yields the edge (001, 010) in this edge grammar:

$$G \rightarrow S_0 \rightarrow (0,\lambda)N_0 \rightarrow (0,\lambda)(0,0)N_0 \rightarrow (\lambda,0)(0,0)(1,1)N_0$$

$$\rightarrow (0,\lambda)(0,0)(1,1)(\lambda,0) = (001, 010)$$

Since both labels in (001, 010) have length 3, (001, 010) is an edge in E_3 (in SE_3) and 001 and 010 are vertices in V_3.

The usefulness of edge grammars as a representation for graph families is dependent upon how much information about a graph or graph family we can determine from its edge grammar. In the next section, we focus on decidability properties and develop a correspondence between edge grammars and graph grammars.

SECTION 2: DECIDABILITY RESULTS FOR EDGE GRAMMARS

Edge grammars were originally developed as an automatable representation for graph families as part of a solution to the problem of mapping large-sized instances of parallel algorithms onto small-sized parallel machines [2], [1]. In this context, it is important to resolve decision problems such as the **membership problem:** *Given a graph H and an edge grammar Γ, is H isomorphic to a member of the graph family $G(\Gamma)$?*

The membership problem and other decision problems are critical in assessing the effectiveness of edge grammars as a representation for graphs and graph families. Although we found that the membership problem was undecidable in general, the problem is decidable when there are additional constraints on Γ. For example, if the graph vertex sets V_n of Γ are bounded from below by an unbounded monotone increasing function f, then the membership problem is decidable. (We call such edge grammars **monotone** edge grammars). Since many of the commonly used parallel interconnection architectures can be generated by monotone edge grammars, the undecidability of the general problem has not proved debilitating to the use of edge grammars in the parallel computation application. The following theorem catalogues some useful decidability results on edge grammars.

Theorem

The following questions are **undecidable.**

1) Given a graph H and an edge grammar Γ, is H isomorphic to a member of $G(\Gamma)$?
2) Given a graph H and a Type 3 edge grammar Γ, is H isomorphic to a subgraph of any member of $G(\Gamma)$?
3) Given an edge grammar Γ, are the graphs in $G(\Gamma)$ planar? connected? hamiltonian?
4) Given edge grammars Γ_1 and Γ_2, are the graphs in $G(\Gamma_1)$ isomorphic to the graphs in $G(\Gamma_2)$? Is there a graph isomorphic to a member of $G(\Gamma_1)$ and a member of $G(\Gamma_2)$? (intersection)

The following questions are **decidable.**

5) Given a Type 3 Edge grammar Γ, is $G(\Gamma)$ finite? infinite? empty?
6) If H is a graph, if there exists a monotone, unbounded function f with $|V_n| \geq f(n)$ for all n, and if Γ is not type 0, then is H isomorphic to a member of $G(\Gamma)$?

Proof:

To prove 1), 3) and 4), we reduce arbitrary NLC grammars [7] to edge grammars and use the undecidability results for NLC grammars. An NLC grammar is a graph grammar in which graphs are generated from a set of initial graphs by applying productions from a fixed set. Upon applying a production $A \rightarrow J$ to a graph H, a single node of H labeled by nonterminal A is replaced by graph J, and J is joined to H according to a given connection chart.

Let G be an NLC grammar. We will construct an edge grammar whose graph family is the family L(G) generated by G. We will do this by constructing a Turing Machine which nondeterministically halts with an edge from L(G) on the tape. By the Hierarchy Theorem in section 3, it is then straightforward to construct an edge grammar Γ which yields the edges on the tape and for which $G(\Gamma)=L(G)$.

Given an NLC G, we construct a Turing Machine M which performs the following procedures:

1) Construct M by encoding the productions and connection chart in the TM program. The tape will be divided into 3 sections:

#counter#graph#work area#

The counter is initialized to the maximum number of nodes in an initial graph plus one. The state is initialized to q_1. The graph is initialized to one of the initial graphs in G. Graphs are represented in the middle section as a sequence of pairs $((v,l_v)(w,l_w))$ where (v,w) is an edge in the graph, l_v is the label of v, and l_w is the label of w. The work area is initialized to the empty string.

2) If A is a nonterminal and H is a graph, M simulates the application of production $A \rightarrow H$ by first building subgraph H in the work area. (H is represented as pairs of (vertex,label) pairs analogous to the graph in the graph area). The TM then scans the current graph (in the graph area) and changes the subgraph in the work area according to the connection chart. In the graph area, the interconnection structure of the node labeled with A is also changed. The connected subgraph H in the work area is unioned

with the modified graph in the graph area yielding a new current graph (in the graph area). The state is reset to q_1 and another production can be applied to the new current graph.

Throughout this procedure, the counter is used to create new nodes. When H is constructed in the work area, the counter contains the unary number for the next new vertex. When a new vertex is created, its vertex number becomes the number in the counter, and the counter is incremented with the number for the next new vertex.

3) After the application of any production in a derivation, the Turing Machine may go into state q_2 and check if the label of every vertex is a nonterminal in the NLC grammar. If so, the Turing Machine enters state q_3, otherwise, it may re-enter q_1.

4) In state q_3, we may assume that the graph has only terminal labels. The workspace can then be erased so that the tape looks like

$$\#counter\#graph\#$$

At this point, M is ready to nondeterministically choose one of the edges in the NLC graph and convert it into an edge for an edge grammar graph. The TM scans the graph area, nondeterministically chooses an edge and copies the edge to the right of the graph area.

$$\#counter\#graph\#((v,l_v),(w,l_w))$$

The TM then encodes the *counter#graph* section of the tape into a unique unary number N and replaces this section with this number.

$$\#N\ (\text{in unary})\#((v,l_v),(w,l_w))$$

Finally, M replaces the current contents of the tape with the edge (x,y) where $|x|=|y|$, $x=1^v0^{N-v}$, and $y=1^w0^{N-w}$. After returning the tape head to an initial position, the TM halts. The tape now contains a terminal pair whose left and right coordinates have equal length.

5) By the Hierarchy Theorem (section 3), it is straightforward to construct an edge grammar which generates (x,y). Each NLC graph in L(G) will now appear as a member of the graph family generated by the edge grammar Γ which simulates M, i.e. $L(G)=G(\Gamma)$.

(Note that the TM will not halt if the terminal graph family generated by the NLC grammar G is empty. However this can be decided in advance and if L(G) is empty, it is trivial to construct an edge grammar whose graph family is also empty).

To show that 1), 3) and 4) are undecidable, it is now sufficient to observe that these problems are undecidable for arbitrary NLC grammars [7] and hence by our construction, for edge grammars.

To prove 2), we first show that the question *"Given a Type 3 edge grammar Γ, does any member G_π of $G(\Gamma)$ contain a self-loop?"* is undecidable. This is a straightforward reduction from the Post Correspondence Problem. Let C be a correspondence system with pairs $\{(u_i,v_i)\}$. Construct a Type 3 grammar Γ with productions

$G \rightarrow (u_i,v_i)G$
$G \rightarrow (u_i,v_i)$

for each i. Then G contains a self-loop iff C has a match. Since the Post Correspondence Problem is undecidable, the self-loop problem is undecidable. To show that the subgraph problem is undecidable, note that if we could determine for a given graph H and a given Type 3 edge grammar Γ whether H was isomorphic to a subgraph of $G(\Gamma)$, then we could instantiate H to be a self-loop and decide the self-loop problem.

To prove 5), note that given a Type 3 edge grammar Γ, the vertex set V_Γ of Γ is contained in the family of context-free languages (see the Hierarchy Theorem in Section 3). Since finiteness, emptiness, and infiniteness are decidable for CFL's, they are decidable for V_Γ. 5) follows from observing that if V_Γ is finite, infinite or empty, then $G(\Gamma)$ must also be finite, infinite or empty. (Note that even if V_Γ is infinite, $G(\Gamma)$ may still be isomorphically equivalent to a finite set of graphs).

To prove 6), we use a simple counting argument. Let f be a monotone, unbounded function such that $|V_n| \geq f(n)$ for all n. Let H be a graph and let $|V_H|=m$. Since f is unbounded and monotone, there exists an N such that $f(n)>m$ for all $n>N$. Test all G_n with $n \leq N$ for isomorphism with H. We can generate each G_n since V_n is computable on an Linear Bounded Automata (see section 3). If some one of these G_i is isomorphic to H, then H is isomorphic to a member of $G(\Gamma)$; if not, then H is not isomorphic to a member of $G(\Gamma)$.

In the same way that conventional grammars form the Chomsky Hierarchy, the graph families generated by Type 0, Type 1, Type 2 and Type 3 edge grammars also form a hierarchy. In the next section, we describe the relationship between the edge grammar hierarchy and the conventional Chomsky Hierarchy.

SECTION 3: EDGE GRAMMARS AND THE CHOMSKY HIERARCHY

To compare the edge grammar hierarchy to the Chomsky hierarchy, we need to compare the same types of objects. Conventional (Chomsky) grammars generate languages whose members are single strings. The language of an edge grammar Γ is a graph family $G(\Gamma)=\{G_n\}$ each of whose graphs G_n consists of a set of edges E_n and their length n incident vertices V_n. We can compare the edge grammar and Chomsky hierarchies by comparing the vertex sets of graphs in the edge grammar hierarchy with languages in the Chomsky hierarchy.

Let Γ be an edge grammar and let $V(\Gamma)$ denote the set of all vertex labels in the graphs of $G(\Gamma)$, i.e. $V(\Gamma) = UV_n$ where for each n, V_n is the vertex set of G_n. For X=0,1,2 or 3, let VX = {$V(\Gamma)$| Γ is a Type X edge grammar}. In other words, we let VX denote the class of all vertex sets for Type X edge grammars. Let (D)LX be the class of (deterministic) languages generated by Type X Chomsky grammars. The following theorem relates the vertex sets VX of the edge grammar hierarchy with the languages LX of the Chomsky hierarchy.

Hierarchy Theorem

a) $L3 \subset V3 \subset L2 \subset V2 \subseteq V1 \equiv L1 \subset V0 \equiv L0$.

b) V3 is incomparable with DL2.

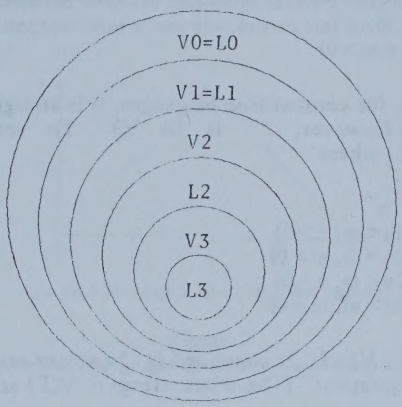


Figure 2
Relationship of the Chomsky and edge grammar hierarchies.

Proof:

Berman in [1] showed $L3 \subset V3$, $L2 \subset V2$, $V1 \equiv L1$, and $V0 \equiv L0$. $V1 \subset V0$ follows from the demonstration in [6] that $L1 \subset L0$. $V2 \subseteq V1$ is true by definition.

To show $V3 \subset L2$, let Γ be a Type 3 edge grammar. Assume without loss of generality that for every production $A \rightarrow (a,b)B$ in Γ, $|a|=0$ or 1 and $|b|=0$ or 1. We will construct a PDA M which accepts vertices in V_Γ. M takes as input a string w and nondeterministically chooses a derivation of w as the left coordinate of a terminal pair or as the right coordinate of a terminal pair. During the derivation, the stack is used to keep track of the disparity in lengths between the left and right coordinate strings of the currently derived terminal pair. More specifically, let Γ be an edge grammar. Construct a PDA M with the following productions:

$(S,\lambda,\lambda) \rightarrow (G_L,\lambda)$
$(S,\lambda,\lambda) \rightarrow (G_R,\lambda)$
$(A_L,a,y) \rightarrow (B_L,x)$, $(A_R,b,y) \rightarrow (B_R,x)$
where $A \rightarrow (a,b)B$ is a production in Γ and

if $|a|=0$, $|b|=1$, and $y=+$, then $x=++$
if $|a|=0$, $|b|=1$, and $y=-$, then $x=\lambda$
if $|a|=1$, $|b|=0$, and $y=+$, then $x=\lambda$
if $|a|=1$, $|b|=0$, and $y=-$, then $x=--$
if $|a|=|b|$, then $x=y$

$(A_L,a,y) \rightarrow (f,x)$, $(A_R,b,y) \rightarrow (f,x)$
where $A \rightarrow (a,b)$ is a production in Γ and x and y are as given above.

It is straightforward to show that M nondeterministically simulates the derivation of a string as the right or left coordinate in a terminal pair and accepts with an empty stack and in the final state only those strings in V_Γ. Proper containment of V3 in L2 comes from showing that V3 and DL2 are incomparable.

To show that V3 and DL2 are incomparable, we show that V3 is not contained in DL2, and DL2 is not contained in V3 by techniques similar to those found in [5]. To show that V3 is not contained in DL2, consider the language

$$L=\{a^n b^m a^{2n} b^{2m} \mid n,m>0\}.$$

By the pumping lemma for context-free languages, it is straightforward to show that L is not in L2. However, L^- is in V3. To see this, observe that $L^-=L_1 \cup L_2 \cup L_3 \cup L_4 \cup L_5$ where

$$L_1=\{a,b\}^*-a^+b^+a^+b^+$$
$$L_2=\{a^n b^+ a^m b^+ \mid 2n<m,\ n>0\}$$
$$L_3=\{a^n b^+ a^m b^+ \mid 2n>m,\ m>0\}$$
$$L_4=\{a^+ b^n a^+ b^m \mid 2n<m,\ n>0\}$$
$$L_5=\{a^+ b^n a^+ b^m \mid 2n>m,\ m>0\}$$

L_1 is in L3 and hence in V3. L_2 is contained in the vertex set $V(\Gamma)$ generated by the following Type 3 edge grammar. (The other strings in $V(\Gamma)$ are in L_1).

G→ (λ,a)G'	G'→ (λ,a)G'
G'→ (a,aa)A	A→ (a,aa)A
A→ (b,aa)B	B→ (b,a)B
B→ (aa,a)C	C→ (aa,a)C
C→ (aa,a)D	D→ (b,a)D
D→ (b,a)	

By slightly modifying the edge grammar given above, it is easy to construct Type 3 edge grammars whose vertex sets contain L_3, L_4, L_5 and parts of L_1. Since V3 is closed under union, $L^-=\cup L_i$ is in V3. Assume towards a contradiction that L^- is also in DL2. Then the complement of L^-, L, must also be in DL2. But L is not even in L2. Hence L^- cannot be in DL2, and V3 cannot be a subset of DL2.

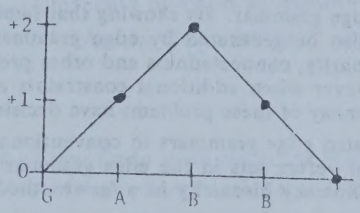
We next show that DL2 cannot be contained in V3. Let $L = \{a^n b^n\}$. L is in DL2. We show that L is not in V3. Assume towards a contradiction that Γ is a Type 3 edge grammar whose vertex set is L. Then all terminal pairs of Γ must be self-loops of the form $(a^n b^n, a^n b^n)$. We show that L is not in V3 by showing that any "long-enough" derivation can eventually pump the left and right coordinates producing strings not in L. Let N be the number of nonterminals in Γ. Consider a derivation of the terminal pair $(a^M b^M, a^M, b^M)$ where $M=N^2$.

Assume the derivation contains a loop $A \rightarrow {}^*(\alpha_2,\beta_2)A$ whose net coordinate difference is zero ($|\alpha_2|=|\beta_2|$) i.e.

$$G \rightarrow {}^*(\alpha_1,\beta_1)A \rightarrow {}^*(\alpha_1\alpha_2,\beta_1\beta_2)A \rightarrow {}^*(\alpha_1\alpha_2\alpha_3,\beta_1\beta_2\beta_3)=(a^N b^N, a^N b^N).$$

Then this loop can be pumped in the same way looping derivations in regular languages can be pumped. In our context, the result will be a terminal pair (w,w) such that w is in $V(\Gamma)$ but not in L. This contradicts the assumption that L is the vertex set of a Type 3 edge grammar.

Now assume that the derivation of $(a^{M_b M},a^{M_b M})$ contains no loop whose net coordinate difference is zero. The derivation can be visualized as a graph in which the horizontal axis is labeled by the nonterminals at each step of the derivation sequence with the start symbol at the origin (Figure 3). The vertical axis gives the net difference $|x|-|y|$ of the left and right coordinates of the currently derived pair (x,y). For the final terminal pair to be in $V(\Gamma)$, the graph must start and stop with difference 0. We can assume without loss of generality that after each application of a production (at each nonterminal along the horizontal axis), the height of the graph changes by +1, -1 or 0.



$G\rightarrow (a,\lambda)A\rightarrow (ab,\lambda)B\rightarrow (ab,a)B\rightarrow (ab,ab)$

Figure 3

A derivation and the graph of its net coordinate differences.

Consider the first loop $A\rightarrow {}^{*}(x_2,y_2)A$ in the derivation of $(a^{M_b M},a^{M_b M})$. Assume without loss of generality that the difference in the x and y coordinates is positive (the case is analogous if the first loop is negative). Since the derivation has length N^2, there are no zero-difference loops and the net difference must end up to be zero, there must be a negative loop $B\rightarrow {}^{*}(x_4,y_4)B$ in the derivation somewhere after the positive loop. We can then rewrite the derivation of $(a^{M_b M},a^{M_b M})$ as follows

$$G\rightarrow {}^{*}(x_1,y_1)A\rightarrow {}^{*}(x_1x_2,y_1y_2)A$$

$$\rightarrow {}^{*}(x_1x_2x_3,y_1y_2y_3)B\rightarrow {}^{*}(x_1x_2x_3x_4,y_1y_3y_4)B\rightarrow {}^{*}(a^{M_b M},a^{M_b M})$$

where $(a^{M_b M},a^{M_b M})=(x_1x_2x_3x_4x_5\ ,\ y_1y_2y_3y_4y_5)$. Let $M=|x_2|-|y_2|$ and $Q=|x_4|-|y_4|$. By assumption, $M>0$ and $Q<0$. Let n be a positive integer. Consider the pair

$$(w_1,w_2)=(x_1x_2(x_2)^{|Q|n}x_3x_4(x_4)^{Mn}x_5\ ,\ y_1y_2(y_2)^{|Q|n}y_3y_4(y_4)^{Mn}y_5).$$

Note that $|w_1|=|w_2|$ and the derivation for $(a^{M_b M},a^{M_b M})$ can be pumped to yield a derivation for (w_1,w_2). Hence (w_1,w_2) is in $V(\Gamma)$. But w_1 and w_2 are not in L since by pumping, there are either too many alternations of a's and b's or the a's and b's are unbalanced. Hence DL2 cannot be contained in V3.

SUMMARY

Edge grammars were first introduced as an automatable representation for graph families in the context of parallel computation. They have proved useful not only for this application but also as a general system for generating many commonly used graphs and graph families which cannot be defined by existing graph grammars and graph generating systems.

In this paper, we have developed a solid context for edge grammars by relating them to conventional formal language theory and graph grammars. We explored the decidability of determining if a given graph is isomorphic to a member or a subgraph of a member of a given edge grammar. By showing that families of graphs generated by NLC grammars can also be generated by edge grammars, we showed that the membership problem, planarity, connectedness and other problems were undecidable for edge grammars. However when additional constraints are made on edge grammars (e.g. monotonicity), many of these problems have decision procedures.

In Section 3, we related edge grammars to conventional formal languages. We showed that the classes of vertex sets in the edge grammar hierarchy fit within the language classes of the Chomsky hierarchy in a "sawtoothed" fashion. This is illustrated in Figure 2.

ACKNOWLEDGEMENTS

We would like to thank Larry Synder and Art Sorkin for their interest and encouragement in this work.

BIBLIOGRAPHY

[1] Berman, F., "Edge Grammars and Parallel Computation," Proceedings of the 1983 Allerton Conference, Urbana, Illinois.

[2] Berman, F. and L. Snyder, "On Mapping Parallel Algorithms into Parallel Architectures," Proceedings of the 1984 International Parallel Processing Conference, Bellaire, Michigan.

[3] Claus, V., Ehrig, H. and G. Rozenberg, Lecture Notes in Computer Science 73: *Graph Grammars and Their Application to Computer Science and Biology*, Springer-Verlag, 1979.

[4] Ehrig, H., Nagle, M. and G. Rozenberg, Lecture Notes in Computer Science 153: *Graph Grammars and Their Application to Computer Science -- 2nd International Workshop*, Springer-Verlag, 1983.

[5] Fischer, P., "Turing Machines with Restricted Memory Access," Information and Control 9, p. 364-370, 1966.

[6] Hopcroft, J. and J. Ullman, *Introduction to Automata, Theory, Languages and Computation*, Addison-Wesley, 1979.

[7] Janssens, D. and G. Rozenberg, "Decision Problems for Node Label Controlled Graph Grammars," JCSS 22, p. 144-177, 1981.

[8] Parker, D., "Notes on Shuffle-Exchange Type Switching Networks," IEEE Transactions on Computers C-29, p. 213-222, March, 1980.

[9] Stone, H., "Parallel Processing with the Perfect Shuffle," IEEE Transactions on Computers C-20, p. 153-161, February, 1971.

DECIDING LIVENESS FOR SPECIAL CLASSES OF COMMUNICATING FINITE STATE MACHINES

Chung-Kuo Chang, Mohamed G. Gouda and Louis E. Rosier

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

1. Introduction

Consider a network of two finite state machines that communicate exclusively by exchanging messages via two one-directional, error-free, unbounded, FIFO channels. Each machine has a finite number of states and state transitions. Each state transition in a machine is accompanied either by sending one message via the machine's output channel or by receiving one message from its input channel.

Networks of communicating finite state machines are useful in modeling[4], analysis[3,7], and synthesis[6,8,15] of communication protocols and distributed systems. The *analysis problem* for these networks can be stated as follows. "Characterize a class of these networks, and construct an efficient algorithm that can decide for any network in the given class whether its communication satisfies some desirable properties." Most of the work to solve this problem has concentrated so far on properties such as boundedness[2,12,14], freedom of deadlocks[2,12,13], and freedom from unspecified receptions[2,12]. These are all safety properties[8]; i.e. they merely guarantee that nothing bad will happen during the course of communication.

In this paper we solve the same problem for liveness properties of such networks. In particular, we characterize two classes of networks (Class 1-k networks whose communication is known to be bounded by some constant k in one direction, and Class 2 networks, where one machine sends one type of message), and construct efficient algorithms to decide three liveness properties for any network in these classes. Actually, each of the three liveness properties can be stated as requiring that some node in a given machine in the network under consideration be reached infinitely often during the course of communication under some fairness assumption. We characterize three graduated fairness assumptions that lead to three degrees of liveness: weak liveness, liveness, and strong liveness as defined in [6].

Following the introduction, the paper is organized as follows: Networks of communicating finite state machines are defined in Section 2. The liveness properties of these networks are presented in Section 3. Then, efficient algorithms

to decide liveness properties for Class 1-k and Class 2 networks are discussed in Sections 4 and 5, respectively.

The work reported in this paper is restricted to decision procedures that decide liveness for special classes of networks. For techniques to prove (rather than decide) liveness for general networks of communicating finite state machines, we refer the reader to [6,9,10,11].

2. Networks of Communicating Finite State Machines

A *communicating finite state machine* M is a directed labelled graph with two types of edges, namely *sending* and *receiving edges*. A sending (or receiving) edge is labelled -g (or +g, respectively) for some *message* g in a finite set G of messages. One of the nodes in M is identified as its *initial node*, and each node in M has at least one outgoing edge.

Let M and N be two communicating finite state machines with the same set G of messages; the pair (M,N) is called a *network* of M and N. A *state* of network (M,N) is a four-tuple [v,w,x,y], where v and w are nodes in M and N respectively, and x and y are two strings over the message set G. Informally, a state [v,w,x,y] means that the executions of M and N have reached nodes v and w respectively, while the input channels of M and N contain the message strings x and y respectively.

The *initial state* of network (M,N) is $[v_0, w_0, E, E]$ where v_0 and w_0 are the initial nodes in M and N respectively, and E is the null string.

Let s=[v,w,x,y] be a state of network (M,N); and let e be an outgoing edge of node v or w. A state s' is said to *follow* s *over* e iff one of the following four conditions is satisfied:

1. e is a sending edge, labelled -g, from v to v' in M, and s'=[v',w,x,y.g], where "." is the concatenation operator.

2. e is a sending edge, labelled -g, from w to w' in N, and s'=[v,w',x.g,y].

3. e is a receiving edge, labelled +g, from v to v' in M, and s'=[v',w,x',y], where x=g.x'.

4. e is a receiving edge, labelled +g, from w to w' in N, and s'=[v,w',x,y'], where y=g.y'.

Let s and s' be two states of (M,N); s' *follows* s iff there is a directed edge e in M or N such that s' follows s over e.

Let s and s' be two states of (M,N); s' is *reachable from* s iff s=s' or there exist states $s_1,...,s_r$ such that $s=s_1$, $s'=s_r$ and s_{i+1} follows s_i for i=1,...,r-1.

A state of network (M,N) is said to be *reachable* iff it is reachable from the initial state of (M,N).

A network (M,N) is said to be of *Class 1-k* iff for every reachable state [v,w,x,y] of (M,N), $|x|<k$, i.e. the communication from M to N is bounded by k.

A network (M,N) is said to be of *Class 2* iff there exists a message g such that every sending edge in N is labelled -g, i.e. N sends one type of message.

3. Liveness Properties of Networks

A *sequence* of a network (M,N) is an infinite sequence $<s_0, s_1,...>$ of reachable states of (M,N) such that s_0 is the initial state of (M,N), and s_{i+1} follows s_i, i=0,1,... .

A node u in machine M (N) is said to *occur infinitely often* in a sequence $<s_0, s_1,...>$ of (M,N) iff for any integer i there exists an integer $j>i$ such that s_j is of the form [u,w,x,y] ([v,u,x,y]), for some w (v), x, and y.

An edge e in machine M (N) is said to *occur infinitely often* in a sequence $<s_0, s_1,...>$ of (M,N) iff for any integer i there exists an integer $j>i$ such that s_{j+1} follows s_j over e.

Based on the above definitions, we can state that something "good" will occur infinitely often in a network (M,N) by stating that a node in M or in N (identified as being "good" or "useful" by the network designer), is guaranteed to occur infinitely often in every sequence of (M,N). Actually, the node need not occur infinitely often in every sequence. Based on the assumption that both machines will progress fairly, the node need not occur infinitely often in any sequence where one machine progresses in an unfair fashion. Next we define three types of fair sequences.

A sequence q of a network (M,N) is called *weakly fair* iff the following condition is satisfied: For any node u in M or N, if u occurs infinitely often in q, then at least one outgoing edge of u occurs infinitely often in q.

A sequence q of a network (M,N) is called *fair* iff the following two conditions are satisfied for any node u, in M or N, that occurs infinitely often in q:

1. Each outgoing sending edge of u must occur infinitely often in q.

2. If there is an infinite number of states of the form $[u,w_i,x_i,y_i]$ ($[v_i,u,x_i,y_i]$)

in q where message g is the head message in x_i (y_i), and if u has an outgoing receiving edge, labelled +g, then edge e must occur infinitely often in q.

A sequence q of a network (M,N) is called *strongly fair* iff the following condition is satisfied: For any node u in M or N, if u occurs infinitely often in q, then each outgoing edge of u must occur infinitely often in q.

Let (M,N) be a network. A node u, in M or N, is said to be *weakly live* (*live*, *strongly live*, respectively) in (M,N) iff it occurs infinitely often in every strongly fair (fair, weakly fair, respectively) sequence of (M,N).

The following theorem states that node liveness for networks of communicating finite state machines are undecidable in general.

Theorem 1: It is undecidable whether a node u in an arbitrary network (M,N) is weakly live (live or strongly live).

In the next two sections, we discuss procedures to decide for any Class 1-k or Class 2 network whether a given node is weakly live (live or strongly live). But first, we present a result that will prove useful later.

Let q be a sequence of (M,N). Then, q_M (q_N) is the sequence of moves (i.e. edges) made by M (N) in q. Two sequences q and q' of (M,N) are said to *correspond* iff $q_M = q'_M$ and $q_N = q'_N$.

Let R' be the set of all sequences q of (M,N) such that if a state [u,v,x,y] is in q, then $|y|=0$ or 1. Also, let R" be the set of all sequences q of (M,N) such that if a state [u,v,x,y] is in q where $|y|>1$, then no subsequent receive move is made by N in q. The following lemma was essentially proved in [12].

Lemma 1: A sequence q is in R iff a corresponding sequence is in R' U R".

It follows that a node u, in M or N, is weakly live (live, strongly live, respectively) iff it occurs infinitely often in every strongly fair (fair, weakly fair, respectively) sequence in R' U R".

4. Deciding Liveness for Class 1-k Networks

In this section, let (M,N) be an arbitrary Class 1-k network, and let R, R' and R" be defined with respect to (M,N) as in the previous section.

Lemma 2: There is no weakly fair, fair, or strongly fair sequence of (M,N) in R". (The proof is by contradiction.)

The following lemma follows immediately from Lemmas 1 and 2.

Lemma 3: A node in M or N is weakly live (live, strongly live, respectively) in (M,N) iff it occurs infinitely often in every strongly fair (fair, weakly fair, respectively) sequence in R'.

The proof of the following lemma is based on the fact that the number of distinct states in each sequence in R' is bounded by a polynomial in 2^k, m and n, where m (n) is the number of nodes in machine M (N).

Lemma 4: If there exists a weakly fair (fair, strongly fair) sequence in R' in which a node u occurs at most a finite number of times, then there exists another such sequence q such that the sequence of moves (i.e. edges) made by M and N in q is of the form $w_1(w_2)^w$ where both $|w_1|$ and $|w_2|$ are bounded by a polynomial in 2^k, m and n.

From Lemma 4, it is useful to distinguish between the two cases of whether the constant k is a parameter of the problem or is a known fixed value. For example, since all the algorithms to decide weak liveness, liveness and strong liveness in this section run in time exponential in k, the next theorem indicates that this probably cannot be improved in the case where k is a parameter rather than a known fixed value.

Theorem 2: Deciding weak liveness, liveness, and strong liveness for Class 1-k networks is PSPACE-Complete provided that k is a parameter rather than a known fixed value.

Proof: The hardness proof is a reduction from the recognition problem for linear bounded automata and is similar in style to the PSPACE-hardness results given in [12]. The fact that the problem is doable in PSPACE follows directly from Lemma 4.

If k is, however, considered to be a fixed value, the problem's complexity needs to be analyzed with respect to parameters m and n; this is somewhat more involved. If a node u is not weakly live (live, strongly live, respectively) in (M,N), then there must exist a strongly fair (fair, weakly fair, respectively) sequence (as described in Lemma 4) in which u occurs only finitely often. Our approach is to determine if such a sequence exists. In order to do this, we need to categorize, in some sense, the possible sequences q mentioned in Lemma 4. This is done basically by categorizing the possible choices for w_2 with respect to each type of fairness. The categorization is as follows:

weakly fair: If a sequence q whose sequence of moves is of the form $w_1(w_2)^w$ is weakly fair, then w_2 must contain at least one edge from M and one edge from N; i.e. w_2 can be categorized by this pair of edges. (Note that w_2 may belong to one or more categories.)

fair or strongly fair: If a sequence q whose sequence of moves is of the form $w_1(w_2)^w$ is fair (strongly fair), then w_2 can be characterized by the "set" of nodes and edges visited in M and N.

In other words, each of the possible sequences q in Lemma 4 is characterized by a set of nodes and edges in M and N, that satisfies some conditions depending on whether we are interested in weakly fair, fair, or strongly fair sequences. We thus call these sets *solution categories*. Notice that for weakly fair sequences, there are O(m*n) different solution categories, but for fair or strongly fair sequences, there are an exponential (in m and n) number of different solution categories.

Lemma 5: There is an algorithm, that runs in nondeterministic logspace, to check for any given solution category whether there is a sequence that belongs to this category.

Thus, an algorithm to check whether a node is not weakly live (live or strongly live) would be to test each solution category to see whether there is a strongly fair (fair or weakly fair) sequence that satisfies Lemma 4. Consequently, we have almost immediately:

Theorem 3: There is an algorithm, that runs in nondeterministic logspace, to decide whether a node in a Class 1-k network is "not" strongly live, provided that k is a known fixed value. Moreover, the problem is complete for this complexity class.

The theorem immediately yields that strong liveness for Class 1-k networks is co-complete for nondeterministic logspace, and so is decidable in polynomial time.

For the case of fair or strongly fair sequences the problem becomes somewhat harder since there are an exponential number of solution categories to check. In order to solve this problem, in polynomial time, we need to develop fast ways to avoid looking at large numbers of candidate categories. This can be done for strongly fair sequences.

Theorem 4: There is a polynomial time algorithm to decide whether a node in a Class 1-k network is weakly live, provided that k is a known fixed value.

The proof involves showing that the nodes and edges in a solution category for

strongly fair sequences must constitute strongly connected components of M and N. Since the strongly connected components of M and N can be found in O(m + n) time[1], the result follows.

We do not know, at the present time, whether a corresponding result holds for deciding whether a node is not live.

5. Deciding Liveness for Class 2 Networks

In this section, we describe algorithms that decide whether a node is *not* weakly live (live or strongly live) in a Class 2 network. For the ensuing discussion, let (M,N) be an arbitrary Class 2 network, and let u be an arbitrary node in M or N.

The results in this section are derived from similar (although more involved) arguments as those presented in the previous section. In particular, Lemmas 4 and 5 can be appropriately modified for Class 2 networks, but it is more difficult than it was in Section 4.

Theorem 5: There is an algorithm, that runs in nondeterministic logspace, to decide whether a node in a Class 2 network is "not" strongly live.

Theorem 6: There is a polynomial time algorithm to decide whether a node in a Class 2 network is weakly live.

Once again we do not know, at the present time, whether a result corresponding to Theorem 6 holds for deciding whether a node is live. We can, however, show a direct correspondence between this problem for Class 2 and Class 1-k networks.

References

[1] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, Reading, Mass., 1974.

[2] Brand, D. and P. Zafiropulo, "On communicating finite-state machines," *Journal ACM*, Vol. 30, No. 2, April 1983, pp. 323-342.

[3] Bochmann, G. V., "Finite state description of communication protocols," *Computer Networks*, Vol. 2, 1978, pp. 361-371.

[4] Bochmann, G. V. and C. Sunshine, "Formal methods in communication protocol design," *IEEE Trans. on Commun.*, April 1980, pp. 624-631.

[5] Chow, C. H., M. G. Gouda, and S. S. Lam, "An exercise in constructing multi-phase communication protocols," in *Proc. of SIGCOMM'84 Symposium*, June 1984. pp. 42-49.

[6] Gouda, M. G. and C. K. Chang, "A technique for proving liveness of communicating finite state machines with examples," in *Proc. of the 3rd Annual ACM Symp. on Principles of Distributed Computing*, Aug. 1984, pp. 38-49.

[7] Gouda, M. G. and Y. T. Yu, "Protocol validation by maximal progress state exploration," *IEEE Trans. on Commun.*, Vol. COM-32. No. 1, Jan. 1984, pp. 94-97.

[8] Gouda, M. G. and Y. T. Yu, "Synthesis of communicating finite state machines with guaranteed progress", *IEEE Trans. on Commun.*, Vol. COM-32. No. 7, July 1984, pp. 779-788.

[9] Hailpern, B. T. and S. S. Owicki, "Modular verification of computer communication protocols," *IEEE Trans. on Commun.*, Vol. COM-31, No. 1, Jan. 1983, pp. 56-68.

[10] Misra, J., K. M. Chandy and T. Smith, "Proving safety and liveness of communicating processes with examples," in *Proc. of ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, Aug. 1982, pp. 18-20.

[11] Pnueli, A., "The temporal semantics of concurrent programs," *Theoretical Computer Science*, Vol. 13, 1981, pp. 45-60.

[12] Rosier, L. E. and M. G. Gouda, "Deciding progress for a class of communicating finite state machines," in *Proc. of the Eighteenth Annual Conference on Information Sciences and Systems*, March 1984, Princeton, New Jersey.

[13] Yu, Y. T. and M. G. Gouda, "Deadlock detection for a class of com-

municating finite state machines," *IEEE Trans. on Commun.*, Dec. 1982, pp. 2514-2519.

[14] Yu, Y. T. and M. G. Gouda, "Unboundedness detection for a class of communicating finite-state machines," *Information Processing Letters*, Vol. 17, Dec. 1983, pp. 235-240.

[15] Zafiropulo, P., C. H. West, H. Rudin, D. Brand, and D. Cowan, "Towards analyzing and synthesizing protocols," *IEEE Trans. on Commun.*, Vol. COM-28, No. 4, April 1980, pp. 651-661.

A SYSTOLIC ARCHITECTURE BASED ON ADDERS FOR DIGITAL SIGNAL PROCESSOR

MICHAEL A. HADDAD
IBM CORPORATION
Endicott, New York

P. A. RAMAMOORTHY
UNIVERSITY OF CINCINNATI
Cincinnati, Ohio

ABSTRACT

This paper describes the design of a systolic arrays for a digital FIR based on adders and shifters. Multipliers use is being avoided because of the complexity of their implementation in VLSI and their slow computation speed. The circuit has been simulated and proved working.

INTRODUCTION

Our new design concerns the multiplication of two vectors using only addition cycles in a systolic architecture.

A systolic architecture, according to Kung, is a general methodology for mapping high-level computations into hardware structures. Briefly, the most important properties are:

1. Design using few types of simple cells.
2. Simple and Regular data and control flow.
3. Local interconnections of the Network.
4. Extensive use of pipelining and multiprocessing for a high rate of computation flow.
5. Modular and regular VLSI implementation.
6. Multiple use of input data for high computational throughput without the associated increase in I/O bandwidth.

THE NON-RECURSIVE FILTER

The non-recursive filter is based on the following function:

$$z_n = \sum_{k=0}^{m} x_k \, y_{n-k} \quad \text{with } n=0,\ 1,\ 2,\ldots$$

x_k being the coefficient that characterize the system, y_{n-k} being a flow of updated input data and, z_n being the outputs of the system.

Different designs proposed the use of multipliers as basic building blocks. But multipliers consume a large chip area and require a large computing time.

For our design purpose, we assume that x_n is represented using 2's complement notation. That is, we have:

$$x_k = -x_k^0 + \sum_{i=1}^{B-1} 2^{-i} x_k^{-i} \quad (x \rightarrow x^{b-1} \ \ldots \ x^0)$$

We then obtain a modified expression for z_n as given below:

$$z_n = \sum_{k=0}^{m} y_{n-k} \ \left(-x_k^0 + \sum_{i=1}^{B-1} 2^{-i} \, x_k^i\right)$$

Using the matrix form, we have then:

$$z_n = \begin{bmatrix} y_{n-k} & \ldots & y_n \end{bmatrix} \begin{bmatrix} x_k^{b-1} & x_k^{b-2} & \ldots & x_k^0 \\ \vdots & & & \\ x_0^{b-1} & & & x_0^0 \end{bmatrix} \begin{bmatrix} 2^{-b+1} \\ 2^{-b+2} \\ \vdots \\ 2^{-1} \\ -1 \end{bmatrix}$$

The goal is to implement a systolic array that will realize this circuit.

Based on a partial computation, we obtain a recursive equation for the calculation of z_n which can be realized by one cell type performing the following:

$$z_n^i = y_{n-k}\, x_k^i + 2^{-1} z_n^{i+1} \qquad \text{(Type I)}$$

Add to that, 2 more types:

$$z_n^0 = -x_k^0\, y_{n-k} + z_n^1 \qquad \text{(Type II)}$$

$$z_n = \sum_{k=0}^{m} x_k\, y_{n-k} \qquad \text{(Type III)}$$

(accumulation)

These three cell types are presented in figures 1, 2 and 3, using only adders and shifters:

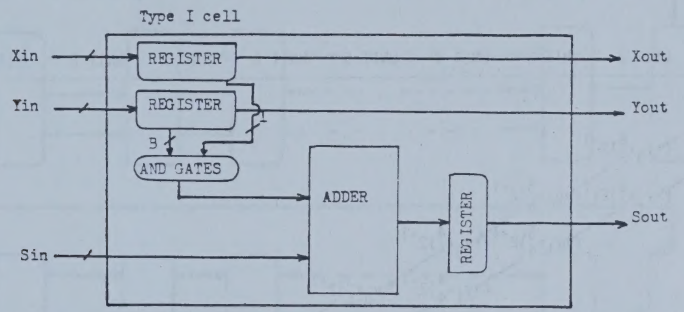


Figure 1

Sout $\leftarrow 2^{-1} * Sin + Yin * Xin^i$
Xout $\leftarrow 2^{-1} * Xin$
Yout $\leftarrow Yin$

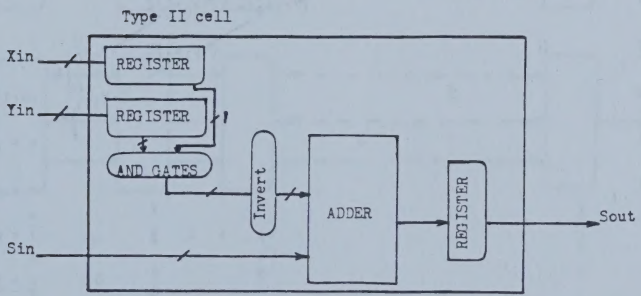
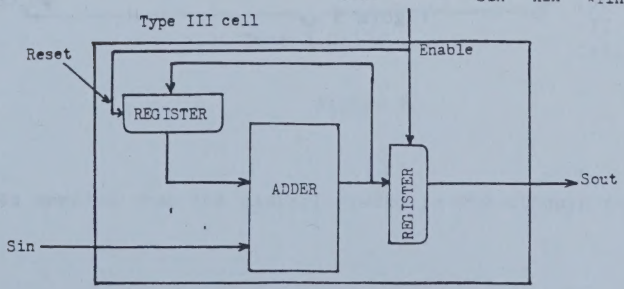


Figure 2

Sout $\leftarrow 2^{-1} * Sin - Xin^B 2^{-1} * Yin$



ACCUMULATION Figure 3

The FIR filter implementation can be accomplished by using B+1 cells in succession to form a stage as shown in Figure 4.

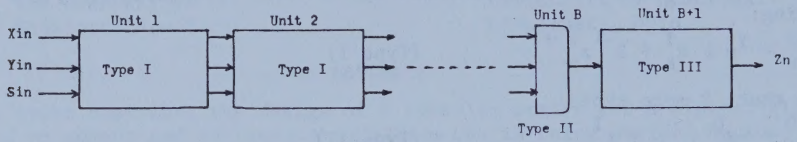


Figure 4

This stage performs its computation as pointed out in Figure 5, to deliver an output. As we see, pipelining is used heavily in the array.

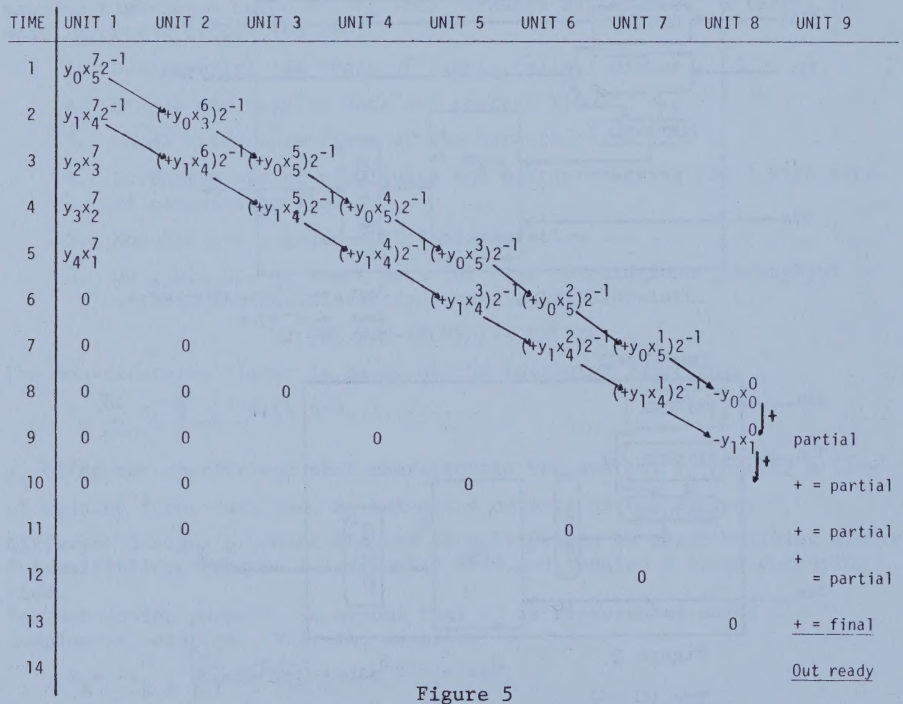


Figure 5

To have an output delivered each clock cycle, we can use the parallelism basics and provide a multiple stages configuration as we see in Figure 6.

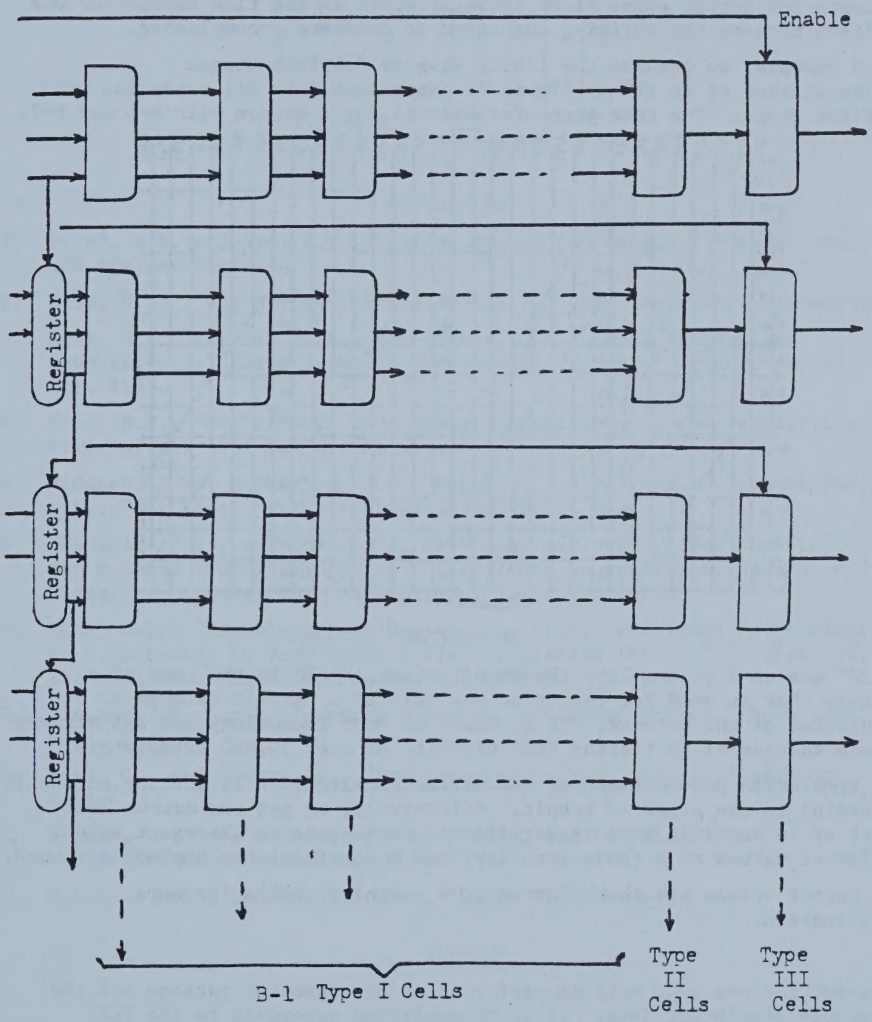


Figure 6

It is to mention that the control system in the circuit is very simple.

TIMING

Timing consideration for this architecture is important because we use extensive pipelining and multiprocessing. In fact, units are working together at any given time and the task of computing the convolution sum is carried out as simple subtasks spread over different cycles. The circuit delivers one output every clock cycle, i.e. it is the time needed for one addition because the shifting operation is hardware accomplished.

As an example, we present the timing diagram for four stages implementation of an FIR in Figure 7. Notations like 2/1 means 2nd unit of first stage. The case presented here is for a system with m=4 and b=7.

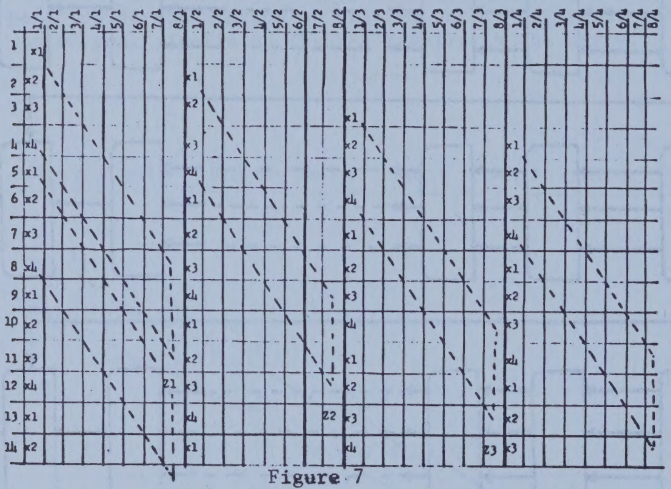


Figure 7

SIMULATION

"RSIM" was used to simulate the architecture. RSIM is the name of a package that is used for real time analysis of a circuit through a simulation of the network. It is based on NMOS technology and was already proven successful in testing VLSI circuits of over 30,000 transistors.

The simulation proved that our FIR filter architecture is working correctly according to the expected result. Effectively, we get one output every clock cycle and this data is verified to correspond to the exact value. Different values of B (data accuracy) and M (coefficients number) are used.

The output values are then plotted on a graphical table for more verification.

LAYOUT

A layout for one type cell is performed using a special package and the validated simulation data. It is accomplished according to the VLSI design rules based on the length unit; that is, the different layers: polysilicon, diffusion, metal, contact and implant, are designed to fit a certain floor plan shown in Figure 8.

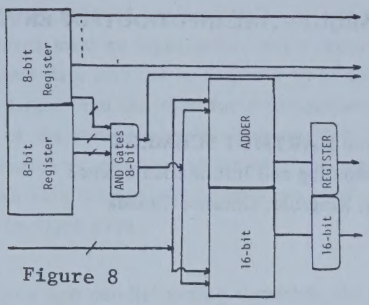


Figure 8

REFERENCES

1. Peled, A.B. and Liu, B.E., Digital Signal Processing - Theory, design, and implementation.

2. Kung, H.T., "Why Systolic Architecutre 7," IEEE Computers, January 1982.

3. Kung, H.T., Sproull, R.F., and Steele, G.L., "VLSI Systems and Computations," Carnegie-Mellon University, Computer Science Press, Oct. 1981.

4. Kung, H.T., "Use of VLSI in Algebraic Computation, Some Suggestions," Proc. of the 1981 ACM Symposium.

5. McCanny, J.V., McWhirton, J.G., Roberts, J.B.G., Day, D.J., and Thorp, T.L., "Bit Level Systolic Arrays," ICASSP83.

6. Cappello, P.R., and Steiglitz, K., "Completely-Pipelined Architecture for Digital Signal Processing," IEEE Trans on Acoustics, Speech, and Signal Processing, Vol. ASSP-31, No. 4, August 1983.

7. Kung, H.T., "Signal-purpose Devices for Signal and Image Processing: An Opportunity in Very Large Scale Integration (VLSI)," 76/SPIE Vol. 241 Real-Time Signal Processing III (1980).

8. Fletcher, W., An Engineering Approach to Digital Design," Prentice-Hall, Inc.

9. Mead, C., and Conway, L., "Introduction to VLSI Systems," Addison Wesley.

10. Terman, C., User's Guide to NET, PRESIM, and RNL/NL," MIT Laboratory for Computer Science.

11. Winston, P.H., and Horn, B.K.P., "LISP" Addison Wesley.

SYSTOLIC SORTING IN A SEQUENTIAL INPUT/OUTPUT ENVIRONMENT

SELIM G. AKL* and HARTMUT SCHMECK**
Department of Computing and Information Science
Queen's University, Kingston, Ontario, Canada

ABSTRACT

A new parallel sorting architecture called the 2-way sorter is presented which has optimal area, period, and time in an environment with sequential input and output, and which meets all the requirements of good algorithms for VLSI.

I. INTRODUCTION

Several parallel sorting algorithms have been proposed lately [1]. These algorithms achieve a significant speedup over sorting on conventional computers. For example, the algorithms of [2], [3], [4] and [5] sort a sequence of n items in $O(\log^2 n)$, $O(n^{1/2})$, $O(\log n)$ and $O(n^e \log n)$, $0 < e < 1$, steps, respectively, on different parallel computational models possessing n/2, n, n log n and n^{1-e} processors, respectively. Clearly, all such algorithms whose running time is sublinear, assume some form of parallel input and output. Otherwise, their asymptotic running time would be dominated by the O(n) sequential steps required to read the input sequence and produce the sorted output, one element at a time. On the other hand, there exist parallel sorting algorithms which do not assume parallel input or output. These algorithms require O(n) steps to sort n items and therefore their asymptotic running time would not be affected should input and output take place sequentially. Examples of such algorithms can be found in [6, page 241], [7], [8] and [9]. In fact, some algorithms are designed specifically so that input and output are performed sequentially [10], [11]. Of particular interest in this paper is the parallel sorting algorithm known as the "odd-even transposition sort" [6, page 241] (see also [12]-[17]), which in its most basic form runs on a parallel computer consisting of n processors interconnected to form a (one-dimensional) linear array.

---

*Work supported by the Natural Sciences and Engineering Research Council of Canada under grant A3336.

**Work supported by the Advisory Research Committee of the School of Graduate Studies and Research of Queen's University under grant 380-420. Current address: Institut fur Informatik und Praktische Mathematik, Christian-Albrechts-Universitat Kiel, D-2300 Kiel, Federal Republic of Germany.

One property of the straightforward implementation of odd-even transposition sort is that the whole sequence to be sorted must be input before any comparison-exchange can take place. For real-time applications where data may not be available all at once, or in cases where the time between two input items is larger than the time for a comparison-exchange, this may result in practice in a slowing down of the sorting process. This problem is addressed in [16] where an efficient on-line sorting algorithm is proposed which overlaps input, comparison-exchange, and output. The algorithm, known as a "compare/steer up-down sort", sorts m sequences of n k-bit items with bit-serial input in (m+1)nk steps.

In this paper we describe a new parallel sorting algorithm, the 2-way sorter, which is particularly suited for an environment with sequential input and output, and which meets all the requirements for a good VLSI implementation [18]: (i) its hardware realization consists of simple processors (or cells) each having a constant number of input/output lines, a constant area and a constant operation time (i.e. independent of the number of cells or of the length of the data items); (ii) the data and control flows are simple, regular and local; (iii) it makes maximal use of pipelining and in every cycle all cells are active (i.e. it makes maximal use of multiprocessing); (iv) it can be easily extended to sort more or larger numbers. Each cell used by the new algorithm operates at the bit level: its only function is to compare two bits and shift one bit. Assuming that the area of such a cell is a and that its operation time is t, then, in order to sort n k-bit numbers, the algorithm requires area $A = nka/2$, period $P = nt$ and time $T = (2n+k-1)t$, if k bits are input at the same time. This leads to an overall complexity of $APT = O(n^3 k)$ which is by a factor of k^2 smaller than the corresponding complexity of the compare/steer bubble sorter [16] or of the "zero-time" sorter [17], which in spite of being a bit-parallel algorithm cannot be fully pipelined, because its comparison-exchanges compare the bits of two numbers from the least significant to the most siginificant bit, i.e. a potential exchange has to wait for the comparison to be completed. The 2-way sorter is optimal with respect to all three complexity measures, if the environment of the sorter can produce and accept k bits (or one number) at a time, i.e. an environment with sequential input and output: Since the last number received as input may be the first to be produced as output, a lower bound on the area is $\Omega(nk)$, the area to store the sequence. The period and the time obviously have to be at least $\Omega(n)$, the time to input or output one sequence of length n. Furthermore, the algorithm can sort m sequences of n k-bit numbers on an area of $(k+1)\,n\,a/2$ in time $((1+\lceil m/2 \rceil)\,n+k)\,t$ using bit parallel input of 2k bits at a time, which essentially is sequential input. This represents a speedup by a factor of 2k over the algorithms of [16] and [17].

II. THE 2-WAY SORTER

In [16] a bit-serial implementation of the up-down sorting algorithm is suggested, using magnetic bubble technology. In this paper we assume that even in a sequential input/output environment all the bits of one number are available at the same time. We suggest a bit-parallel implementation which can be based on very simple processing units or cells and thus qualifies to be a good algorithm for VLSI.

Obviously the time for the bit-serial comparison-exchange of two k-bit numbers $x_{k-1}x_{k-2}\ldots x_o$ and $y_{k-1}y_{k-2}\ldots y_o$ is proportional to k. The bit-parallel comparison-exchange could be done

in time log k at the expense of additional area. Instead we assume a skewed input format and use a comparison-exchange unit which compares pairs of bits of k different pairs of numbers at the same time. Figure 1 shows how this unit can be used for the comparison-exchange of n pairs of numbers $(x^{(i)}, y^{(i)})$, $1 \le i \le n$. The numbers $u^{(i)}$ and $v^{(i)}$ appearing as output satisfy $u^{(i)} \le v^{(i)}$. In the situation shown the comparison-exchange of the first pair has just been completed. As in the bit-serial version the comparison of two numbers needs time k, but the pipelining of numbers with constant period reduces the time for the comparison-exchange of n pairs of k-bit numbers from nk to n+k-1. In the sequel we call this the *skewed bit-parallel* comparison-exchange.

The comparison-exchange unit can be built from cells capable of comparing and exchanging bits. The cells have to pass appropriate information about the result of their comparison from left to right, i.e. from the cell comparing the most significant bits to the cell comparing the least significant bits. The basic structure of these cells is indicated in Figure 2. The two cells shown differ with respect to the ordering they produce: The a-cell is used to produce ascending order (as indicated by the "<" in Figure 2a) and the d-cell is used to produce descending order (as indicated by the ">" in Figure 2b). The reason for introducing these two different types of cells instead of having just one cell type is the simplification of the wiring in the design of the 2-way cell below (see Figure 7). x and y are supposed to be bits at the same position of the pair of numbers to be compared. The inputs r and s provide the control information mentioned above, making sure that the values of u and v will always be the bits of the smaller and the larger number (in the a-cell), or the larger and the smaller number (in the d-cell), respectively.

In the a-cell the values of x and y are exchanged (i.e. u=y and v=x), if r and s are 0 and y<x, or if rs=10. If r and s are 0 then r's' is set to 01, if x<y, and to 10, if y<x. In all other situations we have r's'=rs (see Table 1a). The definition of the d-cell is analogous (see Table 1b). Depending on the desired ordering (i.e. ascending or descending) the a-cells or the d-cells can be combined to form the skewed bit-parallel comparison-exchange unit as shown in Figure 3. For a comparison-exchange r and s have to be set to 0, and after k cycles the values of r' and s' indicate, which of the two numbers has been the larger one, or if they have been equal. Other values for r and s would transform the comparison-exchange unit into an exchange unit or simply a delay unit.

If the structure of the comparison-exchange unit is changed slightly as indicated in Figure 4, we obtain a comparison-exchange register storing a k-bit number (or - more exactly - bits of k numbers) and - if rs=00 - returning the larger of its contents and its input, if a-cells are used and returning the smaller, if d-cells are used. If the control inputs r and s are set to 10 in case of a-cells or to 01 in case of d-cells, the register can be initialized to any desired value. The comparison-exchange register can be used to determine the minimum (a-cells) or maximum (d-cells) of n numbers $x^{(1)},\ldots,x^{(n)}$ by initializing the register to $x^{(1)}$ followed by the input of $x^{(2)},\ldots,x^{(n)}$. A final input enforcing an exchange will produce the minimum (a-cells) or the maximum (d-cells) of $x^{(1)},\ldots,x^{(n)}$ as output.

The comparison-exchange register almost is a bit-parallel version of the compare/steer unit used in the implementation of the up-down sorting algorithm of [16]. The structure of the compare/steer unit is indicated in Figure 5. The unit holds two numbers at C and D initially. In

its "downward" state during each cycle of operation input is provided to A, the contents of D are delivered as output, and the number at C is moved into B. In its "upward" state the input goes to B, the output comes from C, and the contents of D are moved into A. Then the contents of A and B are compared. The minimum is delivered to C and the maximum to D. Thus its output is min(A,B) in its "upward" state and max(A,B) in its "downward" state.

By using a-cells or d-cells the comparison-exchange register can realize these two different functions, although it cannot switch from one function to the other. But as is shown below, a simple change of the structure of the basic cells allows them to simulate the switching between the two different states and, in addition to this, leads to a unit that performs both functions at the same time. The new design for the basic a- and d-cells is shown in Figure 6. The relocation of the input and output of c, r, and s in the d-cell is motivated by its use in the simulation of the "upward" state of the compare/steer unit. In both cells the outputs u' and v' get the same values as u and v, respectively, and we always have c=c'. If c=0, then the inputs x' and y' are ignored, i.e. x and y are supposed to be the bits which have to be compared, leading to the same definition of r', s', u, and v as before. If c=1, then x and y are ignored and x' and y' are the relevant inputs.

The augmented a- and d-cells are used to build a bit-level 2-way comparison-exchange cell as shown in Figure 7. In the sequel we call this the *2-way cell*. If c=0, the 2-way cell simultaneously moves the bit v_1 corresponding to the minimum of two numbers upward and the bit v_2 corresponding to the maximum of two numbers downward. If c=1, the pairs of bits are exchanged between the a-cell and the d-cell, which simulates the switching between the two different states of the compare/steer unit. Hence, whereas the compare/steer unit can only alternate between its two different functions, the 2-way cell can perform both simultaneously. A row of k 2-way cells can be used to simulate the compare/steer unit on inputs of size k, using less area (which still is proportional to k, though), since only two numbers have to be stored, and having a constant period, whereas the period and time of the compare/steer unit are equal and proportional to k, respectively. An (n/2)×k-array of 2-way cells, henceforth referred to as the 2-way sorter and shown in Figure 8, can be used to sort k-bit numbers in several different ways:

(a) Sorting n numbers into ascending order: Assume that initially all the a-cells in the 2-way cells store a 1. Let c be 0 and input the n k-bit numbers at the top, using the skewed input format as mentioned before (see Figure 1). At the bottom constantly input 1's, i.e. the largest k-bit number. After the input of the last number set c to 1 and input an arbitrary number at the top, i.e. the contents of the "downward" and the "upward" columns are switched. c is reset to 0, and in the next n cycles the sorted sequence of numbers appears at the top in ascending order. Since after the switching all the a-cells in the 2-way cells store a 1 again, the next sequence of n numbers may be input, while the sorted first sequence is provided as output. Thus the sorting of two sequences can be overlapped. If we use the same array to sort fewer and/or shorter numbers, we only waste area, since the period and the time adjust to the new problem size. This effect is not achievable by the bit-serial realization suggested in [16] which always expects numbers of the same size. The array can easily be extended to sort larger and/or more numbers by adding columns or rows without having to change the design of the basic cell.

(b) Sorting n numbers into descending order: Assume that initially all the d-cells store a 0. Then the same algorithm as in (a) can be used, but the numbers are input and output at the bottom, and at the top we constantly provide 0's. This algorithm is a dual version of algorithm (a), obviously sorts into descending order, and has the same complexity. Hence we can use the same hardware to produce at optimal speed, period, and area either ascending or descending sequences of numbers.

(c) Sorting two sequences at the same time: In an environment, which can produce and accept two numbers or 2k bits at a time, we can use the systolic array of 2-way cells even more efficiently. The idea is to execute algorithms (a) and (b) concurrently. This is possible, if we attach another leftmost column of 2-way cells, which will get the constant inputs 0 at the top and 1 at the bottom. This will have the same effect of seperating sequences as the use of tag bits in [16], which were necessary there to allow overlapped sorting. The additional column only leads to a slightly larger area without changing the period or the time of the array.

Assume that initially all the a-cells in the modified array contain a 1 and that all the d-cells store a 0. Then we feed two sequences of n k-bit numbers into the array, one at the top and one at the bottom. When c is set to 1, the downward and upward columns exchange their contents. Now every a-cell in the leftmost column contains a 1 and every d-cell in this column contains a 0. Thus, if we feed in the next two sequences at the top and at the bottom, this will not interfere with the parallel output of the first two sorted sequences, because the tag-bits seperate the consecutive sequences.

Theorem 1: The systolic array of 2-way cells can sort m sequences of n k-bit numbers on an area of $n(k+1)a/2$ in time $(([m/2]+1)n+k)t$, where a and t are the area and the time of the - bit-level - 2-way cell. This is optimal, if the bit-level input rate is 2k. □

(d) Sorting large sets of data: If we want to sort larger sequences than the 2-way sorter can handle and if we cannot extend the hardware as indicated above, a strategy for merging sorted subsequences is needed. Assume we have m data chips capable of holding n k-bit numbers each, where n is the capacity of the 2-way sorter. We can either read or write numbers on a chip, but not both at the same time. The first step in sorting these numbers is to sort the numbers on each chip. Using our 2-way sorter, we can do this in time $(([m/2]+1)n+k)t$ as stated in Theorem 1. Assuming that we have no auxiliary data chips available, the m sorted sequences are merged using a generalized version of the odd-even merge algorithm [19].

Theorem 2: Using a 2-way sorter capable of sorting maximally n k-bit numbers, the sorting of mn k-bit numbers can be done in time $O(nm\log^2 m)$ without needing additional memory. □

III. DISCUSSION

It may be worthwile in this context to compare the 2-way sort with two other parallel sorting algorithms recently proposed for implementation by VLSI technology. The first [17] is the "zero-time" sorter, a direct implementation of the algorithm in [16]. As in the 2-way sort the

comparison-exchanges are done in a bit-parallel way. But as already mentioned the comparison-exchanges cannot be pipelined since they are performed one bit position at a time from the least significant bit to the most significant bit. Consequently, in contrast to the 2-way sorter, the "zero-time" sorter still suffers from the same delay proportional to the length of the data items as the bit-serial compare/steer bubble sorter [16]. Furthermore, in every row of the "zero-time" sorter there have to be extra control lines for broadcasting the final result of a comparison to all the cells in the row. This leads to an additional delay in the execution of a comparison-exchange. The "zero-time" sorter extends the design of [16] by allowing to sort sequences input at the top or at the bottom either into ascending or into descending order. But this additional feature results in a more complex mechanism to keep top- and bottom-sequences seperate. The 2-way sorter essentially provides the same choice, because it can sort top- and bottom- sequences at the same time in addition to pipelining the sorting of consecutive sequences. Thus for the 2-way sorter the choice between sorting into ascending or into descending order is a choice between providing the input at the top or at the bottom. The seperation of top- and bottom-sequences is achieved by simply using a tag bit. Summarizing, the 2-way sorter has been shown to have several advantages over the "zero-time" sorter.

The second algorithm is the parallel enumeration sort [11] which uses n processors P_1, P_2,..., P_n to sort the sequence of n numbers x_1, x_2,..., x_n. The processors are connected by a pipeline which allows each P_i to transmit data to P_{i+1}. In addition, two buses are used to which all processors are connected: the input bus allows two processors to receive an input simultaneously, while the output bus allows the processors to produce their outputs one at a time. Each processor contains a p-bit shift register, a p-bit bus register, a comparator for two p-bit numbers, a q-bit counter and a control circuit. All counters are initially set to 0. During the ith time unit, each P_j uses the pipeline to shift the contents of its shift register to the shift register of P_{j+1}. Simultaneously, x_i is loaded into the shift register of P_1 and the bus register of P_i, using the input-bus. Now, all processors compare in parallel the number in their shift register to that in their bus register and, if the latter is larger, increment their counter by 1. After n+i-1 time units, the number in the bus register of P_i, namely x_i, has been compared to all n input numbers and the contents of P_i's counter can be placed on the output bus during time unit n+i. This output, denoted by c_i, is the rank of x_i, i.e. the position of x_i in the sorted sequence. (More precisely, c_i is the rank of x_i minus one.) After 2n time units, all ranks c_1, c_2,...,c_n have been produced and the sequence x_1,x_2,...,x_n can now be sorted in linear time by a conventional (sequential) computer. Compared with 2-way sort, the parallel enumeration sort uses a basic processor which is more complex than the 2-way cell and does not produce a sorted sequence but rather the ranks of the input elements in the sorted sequence. Also, while the bandwidth in bits of the input bus, output bus and pipeline is equal to p, q, and p, respectively, all lines in the implementation of 2-way sort have a 1-bit bandwidth. Furthermore, since the number of processors used by enumeration sort can be at most 2^q, due to the size of the counter, input sequences of length n, where $n > 2^q$, are handled by using $\lceil n/2^q \rceil^2$ copies of the enumeration sorting circuit and $\lceil n/2^q \rceil (\lceil n/2^q \rceil - 1)$ adders. Similarly, since the registers are p bits long, input numbers of length k bits, where $k > p$, are sorted through repetitive use of the same circuit, a solution which results in a substantial increase in the sorting time. These last two approaches are to be contrasted with the ability of the 2-way sorter to be extended indefinitely to deal with longer sequences and larger numbers, by the addition of rows and columns of cells, respectively.

In conclusion, we make the following observation. Very few analyses in the literature (on parallel algorithms in general and on parallel sorting algorithms in particular) take into consideration the time required for data input and output. As pointed out in this paper, and previously in [11] and [16], such considerations are important if an accurate estimate of a parallel algorithm's behavior is to be obtained. The effect of various input and output environments on the running time of a number of parallel sorting algorithms is studied in [20].

REFERENCES

[1] S.G. Akl, Parallel Sorting Algorithms, Academic Press, 1985.

[2] H. S. Stone, "Parallel processing with the perfect shuffle", *IEEE Transactions on Computers*, vol. C-20, pp. 153-161, February 1971.

[3] H.-W. Lang, M. Schimmler, H. Schmeck and H. Schroder, "A fast sorting algorithm for VLSI", Proceedings of ICALP '83, Barcelona, Spain, July 1983.

[4] M. Ajtai, J. Komlos and E. Szemeredi, "An O(n log n) sorting network", Proceedings of the 15th Annual ACM Symposium on Theory of Computing, Boston, Massachusetts, April 1983, pp. 1-9.

[5] S. G. Akl, "Optimal parallel algorithms for selection, sorting and computing convex hulls", to appear in: G. T. Toussaint, Ed., *Computational Geometry*, North-Holland, Amsterdam, 1984.

[6] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, vol. 3, Addison-Wesley, Don Mills, Ontario, 1973.

[7] C. D. Thompson and H. T. Kung, "Sorting on a mesh-connected parallel computer", *Communications of the ACM*, vol. 20, pp. 263-271, April 1977.

[8] J. A. Orenstein, T. H. Merrett and L. Devroye, "Linear sorting with O(log n) processors", *BIT*, vol. 23, pp. 170-180, 1983.

[9] M. Kumar and D. S. Hirshberg, "An efficient implementation of Batcher's odd-even merge algorithm and its application in parallel sorting schemes", *IEEE Transactions on Computers*, vol. C-32, pp. 254-264, March 1983.

[10] S. Todd, "Algorithms and hardware for a merge sort using multiple processors", *IBM Journal of Research and Development*, vol. 22, pp. 509-517, September 1978.

[11] H. Yasuura, N. Tagaki and S. Yajima, "The parallel enumeration sorting scheme for VLSI", *IEEE Transactions on Computers*, vol. C-31, pp. 1192-1201, December 1982.

[12] S. E. Goodman and S. T. Hedetniemi, *Introduction to the Design and Analysis of Algorithms*, McGraw-Hill, Montreal, Quebec, 1977, pp. 272-275.

[13] G. Baudet and D. Stevenson, "Optimal sorting algorithms for parallel computers", *IEEE Transactions on Computers*, vol. C-27, pp. 84-87, January 1978.

[14] T.C. Chen, K.P. Eswaran, V.Y. Lum and C. Tung, "Simplified odd-even sort using multiple shift register loops", *International Journal of Computer and Information Science*, vol. 7, pp. 295-314, 1978.

[15] M.R. Kramer and J. van Leeuwen, "Systolic computation and VLSI", Technical Report RUU-CS-82-9, Vakgroep Informatica, Rijksuniversiteit Utrecht, June 1982.

[16] D. T. Lee, H. Chang and C.K. Wong, "An on-chip compare/steer bubble sorter", *IEEE Transactions on Computers*, vol. C-30, pp. 396-405, June 1981.

[17] G. Miranker, L. Tang, and C. K. Wong, "A "Zero-Time" VLSI Sorter", *IBM Journal of*

Research and Development, vol. 27, no. 2, pp. 140-148, March 1983

[18] M. J. Foster and H. T. Kung, "The design of special purpose VLSI chips", *Computer Magazine*, vol. 13, pp. 26-40, January 1980.

[19] K. E. Batcher, "Sorting networks and their applications", Proceedings of the AFIPS 1968 Spring Joint Computer Conference, vol. 32, April 1968, pp. 307-314.

[20] H. Schmeck, Manuscript in preparation.

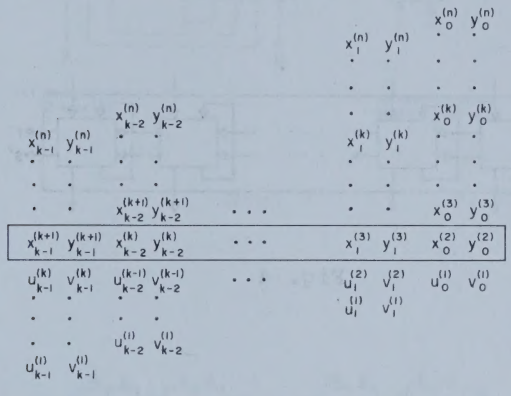


Fig. 1

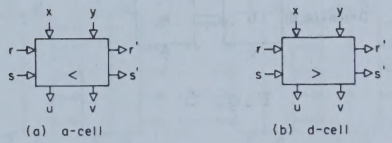


Fig. 2

r s	x y	r's'	u v
r s	0 0	r s	0 0
r s	1 1	r s	1 1
0 0	0 1	0 1	0 1
0 0	1 0	1 0	0 1
0 1	x y	0 1	x y
1 0	x y	1 0	y x

(a) a-cell

r s	x y	r's'	u v
r s	0 0	r s	0 0
r s	1 1	r s	1 1
0 0	0 1	0 1	1 0
0 0	1 0	1 0	1 0
0 1	x y	0 1	y x
1 0	x y	1 0	x y

(b) d-cell

Table 1

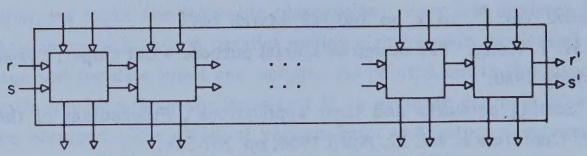


Fig. 3

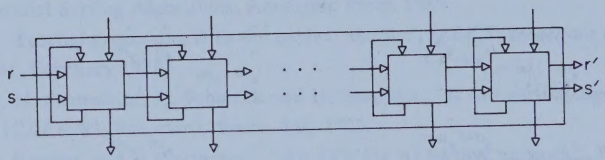


Fig. 4

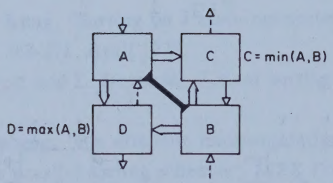


Fig. 5

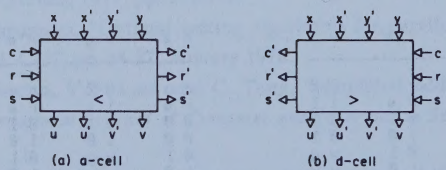


Fig. 6

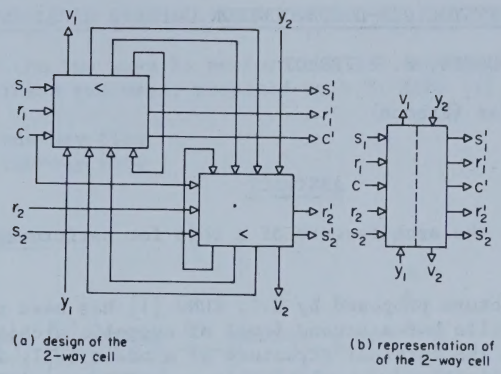

v_1
y_2
s_1
r_1
c
s_1'
r_1'
c'
r_2
s_2
r_2'
s_2'
y_1
v_2
(a) design of the 2-way cell
(b) representation of of the 2-way cell

Fig. 7

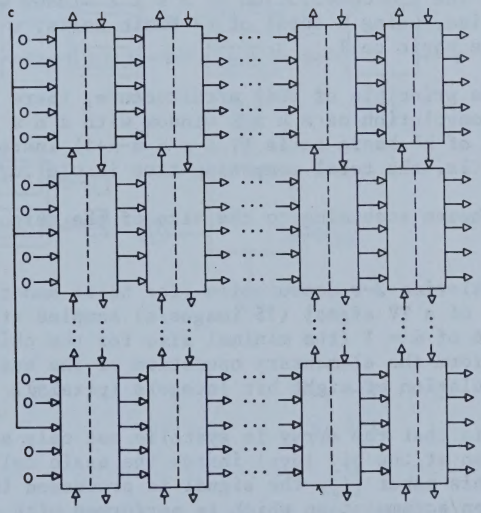

c
0
0
0
0

Fig. 8

A REAL-TIME 2-LEVEL SYSTOLIC 2-D CONVOLUTION CHIP

C. BOZZO, M. FOUQUES, E. SIFFREDI
CSEE-CETIA
La Valette-du-Var (France)

ABSTRACT

This paper describes the architecture of a chip for performing the 2-D convolution.

The systolic architecture proposed by H.T. KUNG [1] has been chosen for the array of basic cells but a second level of systolic architecture has been introduced with the internal structure of a basic cell. The basic cell performs multiplication/accumulation in a serial way with a systolic architecture at the bit level.

The chip can be realized with an ECL gate-array.

1. INTRODUCTION

In one of his papers [1], H.T. KUNG has proposed a systolic architecture to perform on a chip the 2-D convolution of a k x k window with a n x n image in $\sigma(n^2.u/k)$ time, using a total of k^3 basic cells, u representing the cycle time of the basic cell.

In fact, by using the principle of this architecture, there are many ways to perform the 2-D convolution of a k x k window with a n x n image ; by using m kernel cells of k^2 basic cells ($1 \leq m \leq n-k+1$) instead of k kernel cells of k^2 basic cells, the total computing time is $\sigma(n^2.u/m)$.

The term m must be chosen according to the size of the chip and the total computing time.

Interested by the real-time 2-D convolution (for noise smoothing, edge extraction, etc ...) of a TV signal (25 images/s) sampled at 10 MHz ($n = 512$), the choice of $m = 1$ (the minimal size for the chip) leads us to $u \leq 150$ ns to perform the elementary operation of the basic cell, the multiplication/accumulation of eight bit integers (pixels).

The important thing is that the array is systolic not only at the word (pixel) level but also at the bit level inside the basic cell. Indeed, as KUNG's choice in his paper [1], the signal is processed in a serial way for multiplication/accumulation which is performed with a bit-level systolic architecture.

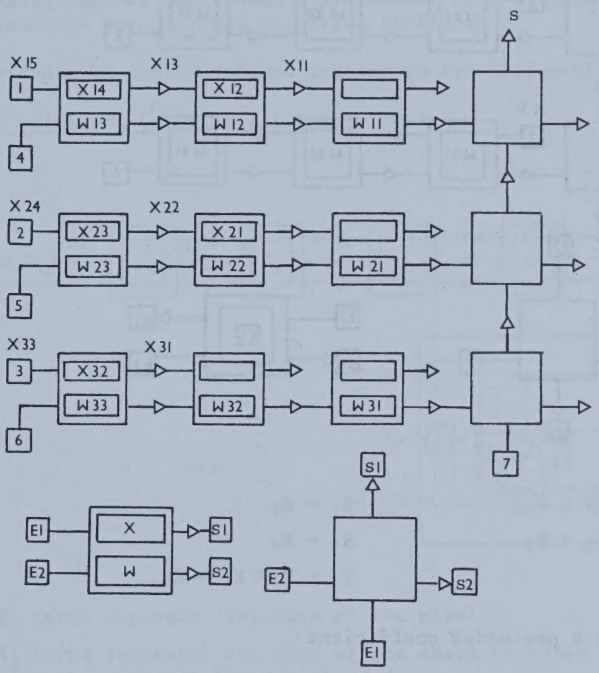
We shall see later that the choice of $k = 3$, $m = 1$ and the serial performing of multiplication/accumulation allows us to realize the chip with a gate-array but leads us to the ECL technology.

2. WORD-LEVED SYSTOLIC ARCHITECTURE FOR 2-D CONVOLUTION

We only resume the two ways to perform the 2-D convolution with a systolic architecture. These two ways, proposed by h.T. KUNG [1] are :

- the one-way flow
- the two-way flow

2.1. One-way flow



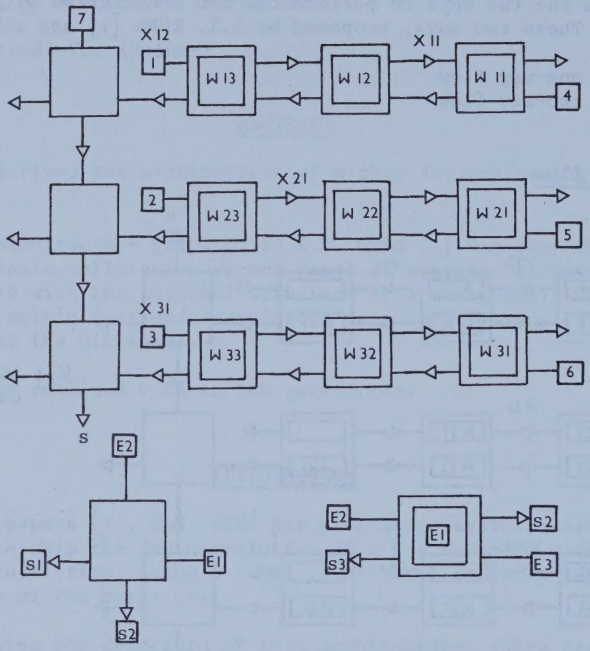
W_{ij} is preloaded coefficient

$S_2 \leftarrow W.E_1 + E_2$ $\qquad$ $S_1 \leftarrow E_1 + E$

$S_1 \leftarrow X$ $\qquad$ $S_2 \leftarrow E_2$

$X \leftarrow E_1$

2.2. Two-way flow



$S_1 \leftarrow E_1$

$S_2 \leftarrow E_1 + E_2$

$S_1 \leftarrow E_1$

$S_2 \leftarrow E_2$

$S_3 \leftarrow E_1 * E_2 + E_3$

W_{ij} is a preloaded coefficient

3. BIT-LEVEL SYSTOLIC ARCHITECTURE FOR MULTIPLICATION/ACCUMULATION

This is a very interesting way to perform the elementary operation of the basic cell.

This architecture has two main advantages :

. it reduces the number of gates
. it reduces the number of pins for packaging
. it simplifies the interconnections

In counterpart, it has an inconvenient :

. it increases the clock frequency of the chip.

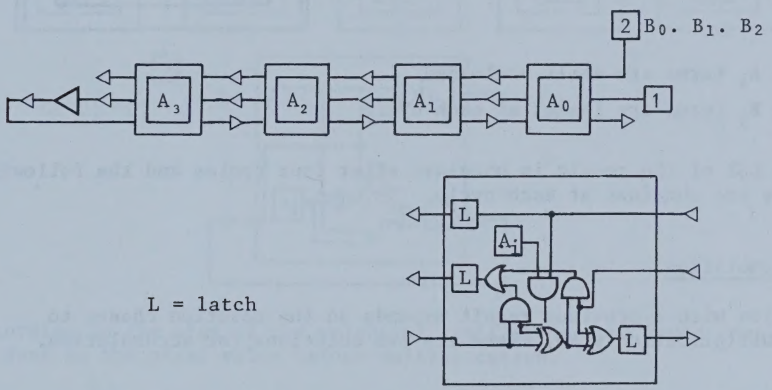
In our application, the word is eight bit long (pixel) and the weighting coefficient is five bit long (4 bits + 1 sign bit). This limitation to five bits, which allows in fact most of applications, has been done to limit the size of the chip but there is no theoretical limit for the length of the weighting coefficient unless the number of gates and the clock frequency for real-time commuting.

3.1. Multiplication

The basic cell is built with 4 microcells (one microcell for each bit of the absolute value of the weighting coefficient).

Two microcells can be defined leading to two different systolic arrays :

First solution (used in a two-way flow)



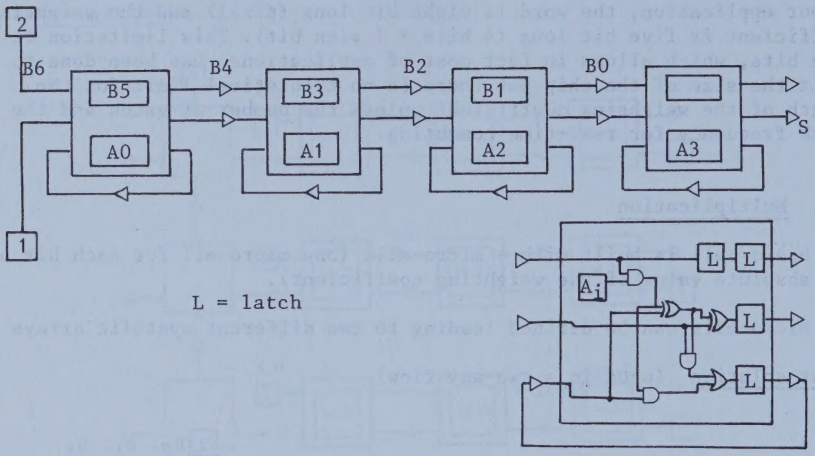
The B_i terms represent the bits of the pixel.

The A_j terms represent the bits of the absolute value of the weighting coefficient, each bit being preloaded.

The B_i terms are loaded at each cycle of a microcell.

The least significant bit of the result is obtained after one cycle and the following bits are obtained at each cycle.

Second solution (used in a one-way follow)



The A_i terms are still preloaded

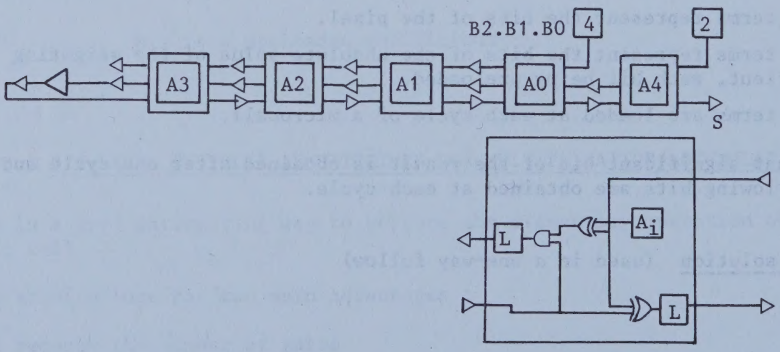
The B_j terms are loaded at each cycle

The LSB of the result is obtained after four cycles and the following bits are obtained at each cycle.

3.2. Accumulation

Accumulation with a previous result depends on the solution chosen to perform multiplication ; so, there are two solutions for accumulation.

First solution



Accumulation is done with two"s complement of the result multiplication result according to the sign of the weighting coefficient.

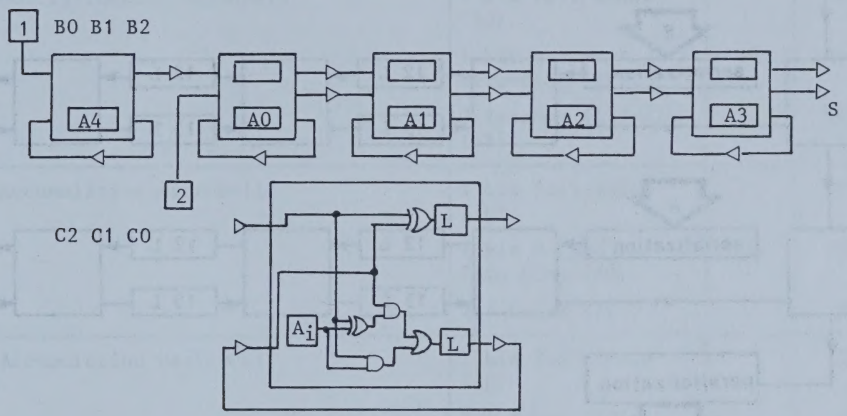
As in the multiplication, the B_j terms are loaded at each cycle.

The C_j terms (bits of a previous result) are loaded at each cycle but with a delay of half a cycle with regard to the B_i terms.

The LSB S_0 of the result is obtained two cycles after the loading of the LSB B_0 of the pixel.

The following bits are obtained at each cycle.

Second solution



According to the sign of the weighting coefficient, the two's complement is done on the pixel value before multiplication.

Then, multiplication and accumulation are performed.

The C_j terms of the previous result are loaded at each cycle but with a delay of one cycle with regard to the B_i terms.

The LSB S_0 the result is obtained five cycles after the loading of the LSB B_0 of the pixel.

The following bits are obtained at each cycle.

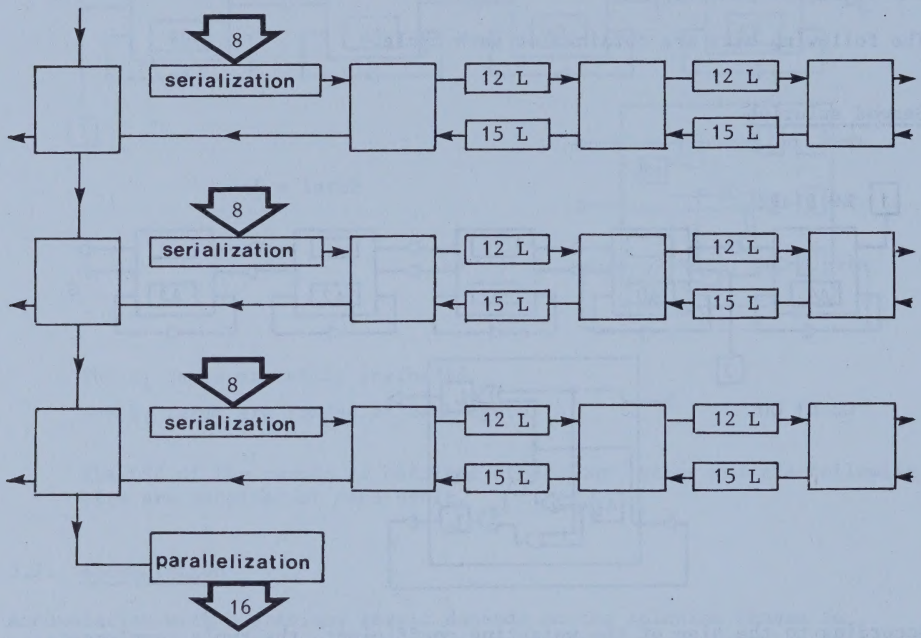
4. THE CONVOLVER ARCHITECTURE

The first solution has been retained for the following reasons :

1) The LSB of the result of a multiplication/accumulation is obtained faster (a two-cycles delay compared with a five cycles delay)

2) The number of gates required is fewer (for one microcell : three latches compared with four latches).

However, the bits of the pixels must be delayed between two microcells to synchronize the pixel and the result ; the delay is 12 cycles long. Note that the result (16 bits long) is not delayed.



The serialization of pixels and parallelization of result has been included in the chip.

So, to have a throughtput rate of 10 MHz, the internal clock must be 160 MHz which imposes ECL technology.

Note that the user sees the chip working on a parallel way at 10 MHz and if a compatibility TTL has been correctly done at the I/O buffers level, the chip doesn't look like an ECL one, unless the specific power supplies and the more important heat dissipation.

4.1. Number of gates required

This paragraph gives an approximation for the size of the chip given in number of gates.

FUNCTION	DEVICE (gates)	NUMBER	TOTAL
Serialization of input data (pixel)	8 bit shift register (80)	3	240
Parallelization of output data	16 bit shift register (160)	1	160
Delay of the pixel between two basic cells	27 latchs (5)	6	810
Multiplication microcell	1 bit full-adder (10)		360
	1 bit D type flip-flop (10)	36	360
	3 latches flip-flop (15)		540
Accumulation microcell	1 bit full-adder (10)		60
	1 bit D type flip-flop (10)	6	60
	2 latches (10)		60
Accumulation basic-cell	1 bit full-adder (10)		30
	2 latches (10)	3	30
			2710

The convolver requires less than 3000 gates and is compatible with the actual state of the art of ECL gate arrays.

The number of I/O pins is few compared with the actual possible packagings.

A TEGAS 5 logical simulation has permitted us to verify the good running of this architecture.

5. CONCLUSION

A new level in systolic arrays has been introduced : the bit-level systolization.

The bit-level systolization has three main advantages :

- the reduction of the number of gates
- the reduction of interconnections
- the reduction of I/O pins

It has an inconvenient :

- the increasing of clock frequency.

In our application, a throughput rate of 10 MHz on parallel data leads us to a 160 MHz clock frequency which imposes to us the ECL technology.

However the reduction of gates allows us to realize the chip with a gate-array.

6. ACKNOWLEDGEMENT

We would like to thank Mr DUMAS, Ph. GRALL and M. SOUFFLET for their help on this work.

REFERENCES

[1] H.T. KUNG, S.W. SONG
A systolic 2 D convolution chip
Carnegie Mellon University - 1981

[2] H.T. KUNG
Why systolic architecture ?
Computer Magazine - 1981

[3] F. ANDRE, P. FRISON, P. QUINTON
Algorithmes systoliques : de la théorie à la pratique
Publication interne n° 196
IRISA - mars 1983

[4] P. QUINTON
The systematic design of systolic arrays
Publication interne n° 193
IRISA - avril 1983

SYSTOLIC ALGORITHMS FOR LOCAL OPERATIONS ON IMAGES

CONCETTINA GUERRA*
Department of Computer Science
Purdue University
West Lafayette, Indiana 47907

Abstract

Local operators, used in many image processing tasks, involve replacing each pixel in an image with a value computed within a local neighborhood of that pixel. Computing such operators at the video rate requires a computing power which is not provided by the conventional computers. Though computationally expensive, local operators are highly regular. Thus a VLSI implementation appears particularly appropriate. This paper presents systolic algorithms for tasks such as connected component determination, distance transform and relaxation, which are defined in terms of local operators.

INTRODUCTION

Typical image processing tasks such as connected component determination, thinning and relaxation are usually carried out by local manipulation, i.e. manipulations where only a few neighbors of each pixel in the input image contribute to determine the value of a corresponding pixel in the output.

Local operators have been applied either in parallel or serial mode. In the serial mode, the image pixels are examined and transformed in some definite sequence, for instance, the forward raster sequence and the updated values of the neighboring pixels are used for each new computation. By contrast, parallel operations involve using the original pixels to compute the values for all pixels.

The form of the local operators suggests an hardware implementation in the VLSI technology. VLSI algorithms have been designed for such tasks on images as convolution [10] and median smoothing [2]. This paper presents two systolic structures which support image processing algorithms defined in terms of local operators applied in parallel or serial mode.

Section 1 reviews the distance transform while section 2 describes two systolic algorithms for such a transform. Section 3 extends the systolic structure to a generic serial local operator. Finally, sections 4 and 5 present tasks such as connected component discrimination and thresholding by relaxation requiring local operators and for which a systolic implementation appears very appropriate.

1. DISTANCE TRANSFORM

The distance transformation was introduced by Rosenfeld [6] mainly as a tool for determining the skeleton of a binary picture , and was extendend thereafter in several ways by different authors.

---

*On leave from Istituto di Automatica, University of Rome, Rome, Italy

We let as usual a digitized image be represented as a rectangular (n,m) array A of pixels a_{ij}. For a binary image, a pixel a_{ij} is said to be external (internal) if it is $a_{ij} = 0 (a_{ij} = 1)$. The distance transform (DT) of A is an array N such that n_{ij} is the minimum distance (see below) of a_{ij} from an external point, if $a_{ij} = 1$, and $n_{ij} = 0$ if $a_{i,j} = 0$. As a definition of distance, one might choose the '4-neighbor' or '8-neighbor' measures considered by Rosenfeld [6] or, say, their generalization to a quasi-euclidean distance such as introduced by Montanari [5] for pictures with an arbitrary number of grey levels. In [5] a hierarchy of reticular approximations to the real plane is introduced (fig. 1). Each intersection point in a reticulum corresponds to a point in the real plane. For binary images, assuming different costs for the elementary edges in the reticulum, the distance between two intersection points P and Q is defined as the lenght of the minimal path through the network. For multivalued images a 'grey-weighted' distance between two points P and Q is defined as the smallest possible sum of grey levels along a path from P to Q.

The distance transformation can be determined by applying an iterative algorihm, which, in the case of black and white pictures, converges in only two iterations. Let us denote with $Q_{i,j}$ the set of neighbors of a pixel a_{ij}. The algorithm uses the following local operation applied sequentially:

$$n_{ij} = \min_{Q^*_{ij}} (n_{ij}, n_{r,s} + d_{ij,r,s}) \qquad (1)$$

where $d_{ij,r,s}$ is the distance between the pixels n_{ij} and $n_{r,s}$ and Q^*_{ij} is the subset of Q_{ij} consisting of the neighbors $n_{r,s}$ of n_{ij} which have already been transformed in the fixed sequence.The algorithm starts with a matrix (n_{ij}) whose element n_{ij} is zero if a_{ij} corresponds to an external point and has a very high constant value otherwise.

For binary images the distance transform involves the application of the sequential operation above only twice, the first time in a forward raster sequence and the second time in a backward raster sequence. Fig. 1 shows the neighbors Q^*_{ij} used in these two scanning sequences: The ones labeled with u's (v's) are used in the first (second) scan of the image. However, in the grey level images, many iterations consisting of forward-backward sequences are required until the image is left unchanged by the procedure.

Alternatively, the DT transform can be accomplished by the following local operation applied to all image pixels in parallel. Notice that in the following expression the entire neighborhood Q_{ij} is considered. $n_{ij}^{(r+1)}$ represents the value of a pixel at $r+1$-th iteration.

$$n_{ij}^{(r+1)} = \min_{Q} \; (n_{ij}^{(r)}, n_{r,s}^{(r)} + d_{ij,r,s}) \qquad (2)$$

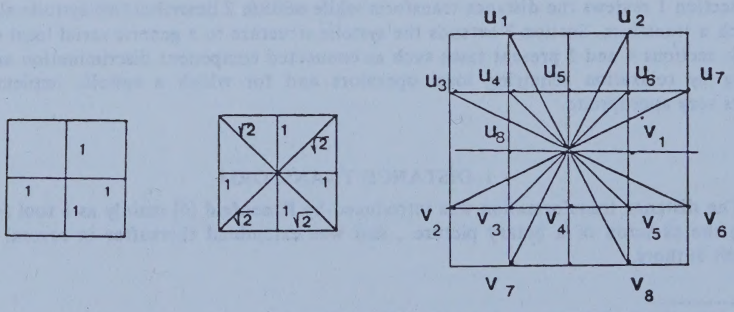


Fig. 1
All numbers represent euclidean length of corresponding side segments

$n+m$ iterations of this parallel operation are required for binary images. The choice between the parallel and sequential algorithm has to be made based on the suitability of the algorithm to the available architecture. A two dimensional array of processors, each one corresponding to a pixel [1] and with a suitable communication geometry, allows each pixel in a processor to communicate with the neighboring ones in a fast and efficient way; furthermore, all processors are executing the same instruction simultaneously on different pixels, driven by a host computer (in SIMD mode). Each iteration of the local operation (2) can be performed on such an array in s clock cycles , s being the cardinality of Q , leading to $s(n+m)$ cyles the necessary time for the entire computation. The sequential approach is obviously more appropriate if a sequential computer is employed. In this case the time is of the order of 2nms' where s' is the cardinality of Q_{ij}^{*}.

We will show in the next section that a better performance can be achieved by the sequential algorithm when both the parallel and sequential algorithms are implemented on a systolic hardware.

2. SYSTOLIC ALGORITHM

This section describes two systolic algorithms for the DT transform. The systolic concept was introduced by Kung [10] mainly to reduce the processor-memory traffic by making multiple use of a data once fetched from the memory. A systolic array consists of a collection of cells with simple communication geometry, regular data flow and intensive pipelining and multiprocessing. Systolic arrays provide cost-effective solutions for a class of image processing tasks, which, like convolution for example, involve repetitive operations on a large set of data.

Consider the DT transform achieved by iteratively applying the parallel operation (2). Letting $s=kxk$ be the size of the considered neighbor Q_{ij} of a generic pixel a_{ij}, the hardware structure which implements the systolic algorithm is an array of s linearly connected cells. Each cell communicates only with its nearest neighbors. the flow of data through the array is similar to the one described in [4] for the 2-D convolution. At each iteration pixels $n_{ij}^{(r)}$ from the image as well as partial results $n_{ij}^{(r+1)}$ move from left to right through the array, one cell at a time. The action of a generic cell is as follows: at each step and for incoming $n_{rs}^{(r)}$ and $n_{ij}^{(r+1)}$ the sum $d_{ij,rs}+n_{rs}^{(r)}$ is computed; the minimum between such a sum and $n_{ij}^{(r+1)}$ is determined to $n_{ij}^{(r+1)}$ and sent to the next cell. We will now describe in detail how the algorithm works. As input to the array, at the $r+1-th$ iteration, we consider the following concatenation of the rows of the image where between elements of two consecutive rows $(k-1)/2$ zero elements are inserted.

$$(n_{11}^{(r)},n_{12}^{(r)},\ldots,n_{1m}^{(r)},0,\ldots,0,n_{21}^{(r)},\ldots,n_{2m}^{(r)},\ldots,n_{n1}^{(r)},\ldots,n_{nm}^{(r)})$$

The results $n_{ij}^{(r+1)}$ are also arranged in a row-by-row fashion with insertion of zeros between consecutive rows:

$$(n_{11}^{(r+1)},n_{12}^{(r+1)},\ldots,n_{1m}^{(r+1)},0,\ldots,0,n_{21}^{(r+1)},\ldots,n_{2m}^{(r+1)},\ldots,n_{nm}^{(r+1)})$$

$n_{ij}^{(r+1)}$'s enter the array with the initial value of $n_{ij}^{(r)}$; are updated as they traverse the pipeline so that by the time they reach the last cell the final value is obtained.

To ensure a correct data flow, the $n_{ij}^{(r+1)}$ have to move twice as fast as the input data $n_{ij}^{(r)}$'s, i.e. each $n_{ij}^{(r)}$ stays in each cell one cycle more. It can be observed, also, that non contiguous elements in the input sequence are involved in the computation of $n_{ij}^{(r+1)}$ so that an appropriate additional delay must be provided for input pixels at certain cells when they move through the array. Let us show that by assuming $k=3$ and $n=5$. Consider the computation of element $n_{22}^{(r+1)}$. Elements $n_{31}^{(r)}$ and $n_{23}^{(r)}$ which both contribute to determine $n_{22}^{(r+1)}$ are in the input sequence $n-1$ distant apart and have to be conveniently delayed to be able to meet $n_{22}^{(r+1)}$ along the array. In fig.3 a few steps of the computation of $n_{22}^{(r+1)}$ are shown. $n_{22}^{(r+1)}$ enters the array with $n_{13}^{(r)}$ thereafter it meets $n_{32}^{(r)}$ and $n_{31}^{(r)}$ in cell 2 and 3 respectively. Notice that each $n_{ij}^{(r)}$ stays in a cell for one cycle. At cell 3 each input pixel is delayed by $n-1$ cycles, and this is obtained by keeping in the cell a stack of $n-1$ elements. That ensures that by the time $n_{22}^{(r+1)}$ leaves cell 3 the value $n_{23}^{(r)}$ is output and enters cell 4 together with $n_{22}^{(r+1)}$. All

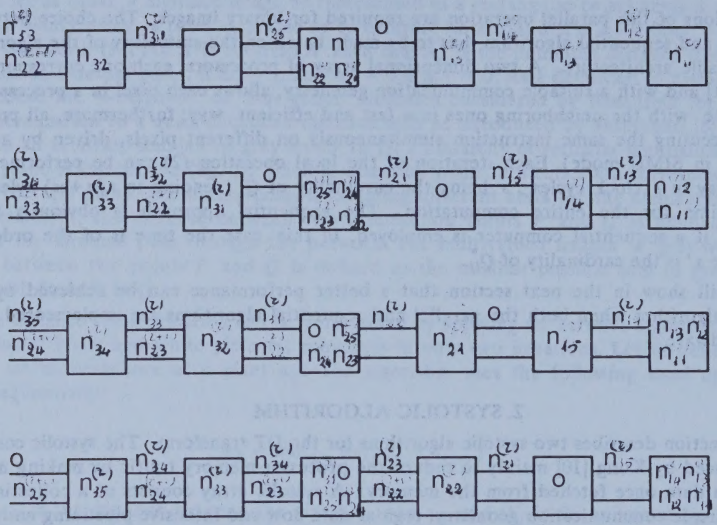


Fig. 3

other cells in the array in positions hk (h=1,2,3) behave like cell 3, that is they have pixels staying inside for n-1 cycles.

In general, for a kxk array input pixels are delayed by $n-1$ at cells hk ($h=1,...,k$) and are delayed by 1 at the remaining cells.

A systolic algorithm for the DT, which uses operation (1) applied sequentially to image pixels, can be implemented on a linear array of s^* cells, where $s^* = (kxk)/2$ is cardinality of Q_{ij}^*. In addition to the ordinary links, the array has a feedback link which allows each newly computed value appearing at the right end cell to reenter the array.

Data move in two opposite directions: $n_{i,j}$'s, initialized to zero, enter the array at the leftmost cell, compute their values as they pass through the array from left to right; the final value, obtained at the rightmost cell reenters the array and then moves from right to left to be used in the computation of neighboring pixels. To ensure a proper data flow $n-k$ zeros are added between elements on consecutive rows of N. Also, consecutive values on input data stream and output data stream are separated 1 clock cycle. As a consequence, only one half of cells are active at each step.

Cells $hk+1$ ($h=1,2,..,(kxk)/2$) have $n-1$ registers to store data which have to be delayed in order to obtain a proper data flow. In addition to that, the rightmost cell keeps each output value one cycle more to be able to use it when, at the next cycle, a new data is entering the cell from the left. The action of a cell is basically the same as in the previous array; the only difference is that the two inputs come from opposite directions. Fig. 4 shows a few steps of the computation for $k=3$ and $n=5$.

Note that with the latter hardware structure the Dt transform is achieved in much reduced time: the two iterations required by the sequential algorithm are performed by having the image data transferred from the external memory to the array and processed only twice; whilst with the parallel approach $n+m$ iterations imply $n+m$ data transfers and processings.

2. A GENERALIZED SYSTOLIC ARRAY

This section presents a systolic array for a generic local operation which involves both pixels a_{ij} from the original image and pixels n_{ij} which have already been transformed. The form of such an operation involving a kxk (k odd) neighborhood is the following:

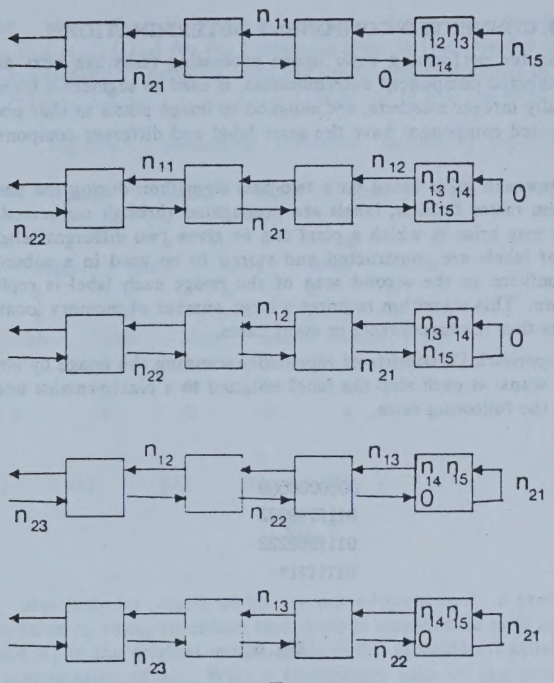


Fig. 4

$$n_{i,j} = f\,(n_{i-p,j-p}, n_{i-p,p+1}, \ldots n_{i-pj+p}, \ldots, n_{i,j-p}, \ldots, n_{i,j-1}, a_{i,j}, \ldots, a_{i,j+1}, \ldots, a_{i+p,j-p}, \ldots, a_{i+p,j+p}) \qquad (3)$$

where $p = k/2$ and the function f is usually defined in terms of arithmetic or boolean operations. In the next section some examples of image processing operations which can be expressed by (3) will be presented.

Basically, the hardware structure consists of two parts: a linearly connected array with one-way pipelining , which performs computation relative to original image pixels, and a linearly connected array with two-way pipelining operating on updated values of pixels. These two arrays are linked at the right end and left end cells respectively. Cell *kxk* has a feedback link to allow a two-way pipelining.(**Fig. 5**)

Input and output pixels enter from the left end of the array; they move through the first *kxk* /2 cells. At central cell *kxk* /2 the original values, no longer useful, disappear whilst the output values continue to move to the right. At cell *kxk* n_{ij} have their final value; at this time they start moving backwards to be used for new computations and finally they reach cell *kxk* /2 where they are output. The data flow in the first half of the array is the same described in the previous section for parallel DT transform; in the second half is like the one for sequential DT.

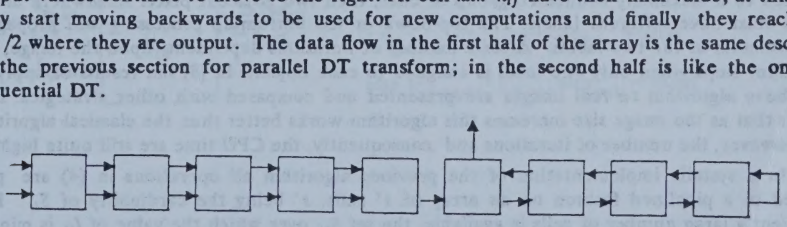


Fig. 5

4. CONNECTED COMPONENT DETERMINATION

Local operations for performing basic image processing tasks are next described. The first of these, the connected component determination, is used to segment a binary image into regions. Labels, usually integer numbers, are assigned to image pixels so that pixels belonging to a maximally connected component have the same label and different components have distinct labels.

The standard approach [6] is based on a two-pass algorithm: during the first scan of the image, in a top-bottom raster fashion, labels are propagated through connected components. Conflicting situations may arise in which a pixel can be given two different labels (see fig.6). Equivalence classes of labels are constructed and stored to be used in a subsequent step to disambiguate such conflicts: in the second scan of the image each label is replaced with its smallest equivalent one. This algorithm requires a large number of memory locations for storing equivalence classes that is unacceptable in many cases.

An alternative approach [3] consists of repeatedly scanning the image by alternating top-down and bottom-up scans: at each step the label assigned to a pixel remains unchanged or is updated according to the following rules.

0000000000
0111000022
0111000222
0111111*

Fig. 6

Step 1. (top-bottom scan): assign to each nonzero pixel a_{ij} a new label l_{ij}

Step 2. (top-bottom scan): assign to each pixel a label which is the minimum of the set of labels including the pixel's own label and the nonzero labels of neighboring pixels already examined in the scan sequence. Letting Q_{ij}^{*} be the set of the updated values of pixels adjacent to the pixel a_{ij} and S_{ij} the subset of Q_{ij}^{*} consisting of nonzero elements, we have:

$$l_{ij} = \text{Min}_{S_{ij}}(l_{rs}, l_{ij}). \tag{4}$$

Step 3. (bottom-top right left scan): like step 2 with a different scan sequence, bottom up instead of top-bottom. Consequently the set S_{ij} over which l_{ij} is at each step will be the symmetric of S_{ij} with respect to rows and columns.

Steps 2 and 3 above are iterated until no changes occur in the image.

It is easily shown that this algorithm performs correctly; if more than one label is assigned to a maximally connected group of ones, then two adjacent pixels somewhere in the group must have different labels. The top-down or the bottom-up processing will propagate the minimum of the two labels. After a number of iterations dependent upon the image, the algorithm stops when only one label is assigned to each object. In [8] the results of applying the above algorithm to real images are presented and compared with other strategies. It is shown that as the image size increases this algorithm works better than the classical algorithm [6]. However, the number of iterations and ,consequently, the CPU time are still quite high.

In a systolic implementation of the previous algorithm all operations in (4) are performed in a pipelined fashion on an array of s' cells, s' being the cardinality of S_{ij}. If a sufficiently large number of cells is available, the set S_{ij} over which the value of l_{ij} is minimized at each step, can be extented to a large neighborhood of l_{ij} without increasing significantly the throughput of the array but reducing at the same time the number of iterations. Let us show that with the aid of an example. Consider the sets

Consider the sets $S_{ij} = \{l_{i-1,j-1}, l_{i-1,j+1}, l_{i,j-1}\}$ for the top-bottom scan and $S^*_{ij} = \{l_{i+1,j-1}, l_{i+1,j}, l_{i+1,j+1}, l_{i,j+1}\}$ for the bottom-up scan; in this case five iterations (fig.7b-f) of the algorithm are needed to properly label the connected component of fig.7a; with the two larger sets $S_{ij} = \{l_{i-1,j-1}, l_{i-1,j}, l_{i-1,j+1}, \ldots, l_{i-1,j+k}, l_{i,j-1}\}$ $(k > 4)$ the same result is achieved in only one pass.

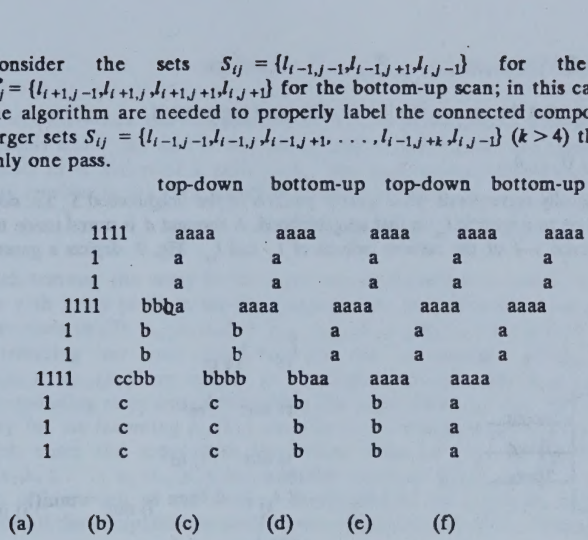
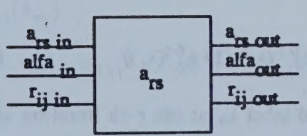


Fig. 7

Since now S_{ij} also includes pixels which are not adjacent to l_{ij} a preliminary test must be done for each value $l_{r,s}$ in S_{ij} to ensure that there is a connected path consisting of only ones between $a_{r,s}$ and a_{ij} in the original image. Only if this condition is satisfied the value $l_{r,s}$ will enter in the computation of l_{ij}. With a preliminary scan of the image, each pixel a_{ij} is assigned a pair of integers r_{ij} and t_{ij} representing the number of ones on the row i to the right and to the left of l_{ij} respectively. Thus at each step to check whether or not l_{ij} and $l_{r,s}$ are connected a single comparison between $s-j$ and r_{ij} is what is needed.

The systolic algorithm, using a two-way pipelining, is next described ; the data flow is the same as described in section 2 and will not be repeated. Thus only the action of a generic cell of the array will be considered here.

step 1. {compute r_{ij} for each pixel}: the image pixels traverse the array and count the number of ones to the right of each pixel. The action of a generic cell is shown in fig. 8.



$a_{rs} = a_{rs\ in}$;

$a_{rs\ out} = a_{rs}$;

if alfa and $(a_{rs} = 1)$ then

$\{r_{ij\ out} = r_{ij\ in} + 1;\ alfa_{out} = alfa\}$

else $\{r_{ij\ out} = r_{ij\ in};\ alfa_{out} = false\}$

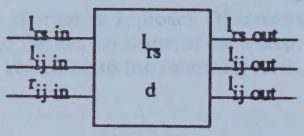
Fig. 8

step 2. {compute t_{ij} for each pixel}

similar to the previous step except that the ones to the left of each pixel are counted

step 3. {compute l_{ij} = Min $_{S_{ij}}(l_{ij}, l_{rs})$}

Each cell of the array logically corresponds to a specific position in the neighborood S_{ij} i.e. computes the minimum (l_{rs}, l_{ij}) relative to a specific l_{rs} in that neighborhood. A constant d is stored inside the cell which represents the difference $s-j$ of the column indeces of l_{ij} and l_{rs}. Fig. 9 depicts a generic cell.



$l_{rs} = l_{rs\ in}$;

$l_{rs\ out} = l_{rs}$;

$r_{ij\ out} = r_{ij\ in}$;

if $r_{ij} > d$ then $l_{ij\ out} = \min(l_{ij\ in}, l)$

else $l_{ij\ out} = l_{ij\ in}$;

Fig. 9.

This step is iterated until no changes occur.

5. RELAXATION

A wide class of problems in image analysis can be formulated in terms of assignments of labels to objects. For example segmentation by thresholding involves a classification of pixels into "light" and "dark" classes. Also interpreting an image implies the assignment of a name to each object present in the image according to a model of the image. This class of problems can be solved by a relaxation technique. Basically, labels are assigned to pixels with initial estimates of their probabilities. These probabilities are iteratively improved by using local constraints between labels. The algorithms are inherently parallel since they use local contextual information to assist the labeling process.

Lets consider the relaxation applied to the segmentation by thresholding [9]. The relaxation process is used to reduce errors in the thresholding operation due to the presence of noise. The initial probabilities for a pixel to be classified as dark and light are estimated in terms of its position along the grey scale. Such estimates are iteratively updated according to the following formula which takes into account not only the pixel grey levels but also the spatial relationships among pixels:

$$p_{ij}^{(r+1)}(\lambda_k) = p_{ij}^{(r)}(\lambda_k)\,(1+q_{ij}^{(r)}(\lambda_k)) / \sum_{t=1,m} p_{ij}^{(r)}(\lambda_t)\,(1+q_{ij}^{(r)}(\lambda_t)) \qquad (5)$$

where $p_{ij}^{(r)}(\lambda_k)$ is the probability that pixel (i,j) is assigned label λ_k at the r-th iteration and where

$$q_{ij}^{(r)}(\lambda_k) = \sum_{rs} \sum_{t=1,m} r_{ij}(\lambda_k \lambda_t)\, p_{rs}^{(r)}(\lambda_t) \qquad (6)$$

In (5) and (6) $m=2$ since λ can anly assume two values corresponding to dark and light. The external sum is computed over a fixed neighborhood of pixel (i,j). $r_{ij,rs}$ are the "compatibility coefficients" and represent how label λ_t at pixel (r,s) influences label λ_k at pixel (i,j). The expression (6) can be rewritten as

$$q_{ij}^{(r)}(\lambda_h) = \sum_{t=1,m} \sum_{rs} r_{ij,rs}(\lambda_h \lambda_t) p_{rs}^{(r)}(\lambda_t). \qquad (7)$$

We will show how the computation (7) can be performed in a systolic way. For m the number of labels and k the size of the considered neighborhood, the systolic structure for relaxation consists of m arrays of k cells each. The computation can be organized as follows: the v-th array receives at the r-th iteration as input the data

$$p_{11}^{(r)}(\lambda_v), p_{12}^{(r)}(\lambda_v), \ldots, p_{nn}^{(r)}(\lambda_v)$$

which traverse the array in the same way as described in section 2 for parallel DT transform. The v-th array perform all the computations involving these values; that is it computes the inner sums in (7), $\mu_{ij}(\lambda_h, \lambda_v) = \sum_{rs} r_{ij,rs}(\lambda_h, \lambda_v) p_{rs}(\lambda_v)$ for a fixed v and for $h = 1,\ldots,m$, thus contributing for one term to all the expressions $q_{ij}^{(r)}(\lambda_h)(h = 1,\ldots,m)$. The sums $\mu_{ij}(\lambda_h, \lambda_v)(p_{rs}(\lambda_v)$ are output at the right end cell and then combined (added) with the corresponding sums coming out from the other linear arrays. More precisely, each cell of the array for an incoming $p_{rs}(\lambda_v)$ contributes to compute all the terms $u_{ij}(\lambda_h, \lambda_v)$ $(h = 1,\ldots,m)$ which enter the array with the initial value of zero in the following order: $u_{ij}(\lambda_1, \lambda_v)$, $u_{ij}(\lambda_2, \lambda_v), \ldots, u_{ij}(\lambda_m, \lambda_v)$; the updated values of $u_{ij}(\lambda_h, \lambda_v)$ $(h = 1,\ldots,m)$ are then sent to the next cell. A cell of the array v corresponding go a specific position in the neighborhood, stores all the compatibility coefficients relative to that point. When output, the u_{ij} are added

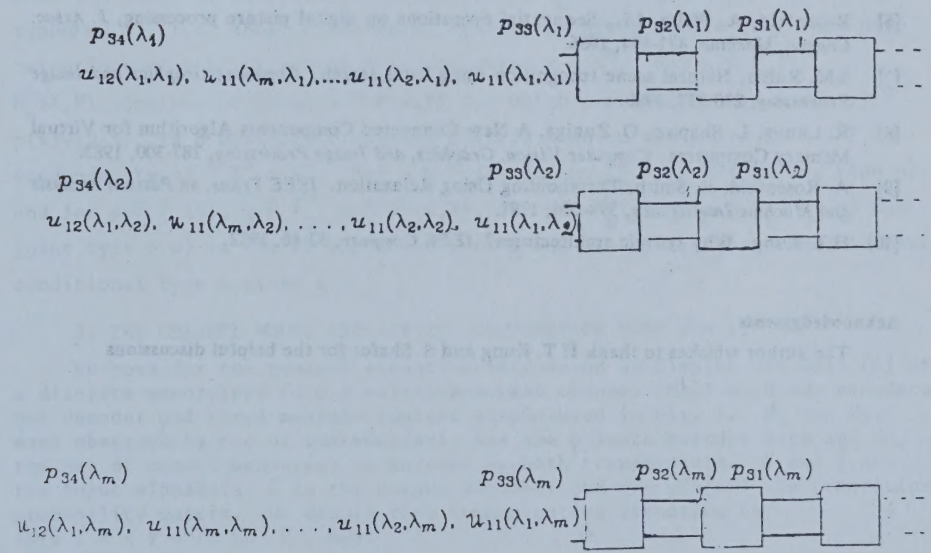


Fig 11

Conclusions

A number of local operators have been reviewed and their implementation on a systolic array has been discussed. In principle, one can expect the parallelization of any task to always convey some advantages when implemented on special hardware consisting of many processors working together to accomplish the same task. Sometimes, however, the identification of such advantages is not quite straightforward. For a number of tasks, including DT transform and connected component determination it is shown that a sequentially applied local operator appear more appropriate than a parallel one when both are executed on a systolic hardware.

References

[1] M.J.B. Duff, Review of the CLIP architecture for image processing. In *AFIPS Conference Proc.*, pages 1055-1060, 1978.

[2] A.L. Fisher, Systolic Algorithms for Running Order Statistics in Signal and Image Processing, H.T. Kung, B. Sproulle, G. Steele (editor), *VLSI Systems and Computations*, pages 265-272. Computer Science Press.

[3] R.M. Haralick, Some neighborhood operators M. Onoe, K. Preston, A. Rosenfeld (editor), *Real-Time/Parallel Computing*, pages 11-34, 1981.

[4] H.T. Kung, L.M. Ruane, D.W. Yen, A two-level Pipelined Systolic Array for Convolutions. H.T. Kung, B. Sproull, G. Steele (editor), *VLSI Systems and Computations*, pages 255-264, 1981.

[5] Montanari, U., A method for obtaining skeletons using a quasi-Euclidean distance. *J. Assoc. Comput. Mach.*, 600-624, 1968.

[6] Rosenfeld, A. Pfaltz, J.L., Sequential operations on digital picture processing, *J. Assoc. Comput. Machine*, 471-494, 1966.

[7] S.M. Rubin, Natural scene recognition using locus search. *Computer Graphics and Image Processing*, 298-322, 1980.

[8] R. Lumia, L. Shapiro, O. Zuniga, A New Connected Components Algorithm for Virtual Memory Computers. *Computer Vision, Graphics, and Image Processing*, 287-300, 1983.

[9] A. Rosenfeld, R. Smith, Thresholding Using Relaxation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 598-606, 1981.

[10] H.T. Kung. Why systolic architectures? *IEEE Compuer*, 37-46, 1982.

Acknowledgments

The author whishes to thank H.T. Kung and S. Shafer for the helpful discussions

CODES GENERATED BY PERMUTATIONS FOR DISCRETE MEMORYLESS MULTIPLE-ACCESS CHANNELS WITH SLEPIAN-WOLF CORRELATED SOURCES

Kristien DE BRUYN★
K.U.LEUVEN, Dept. of Mathematics, B-3030 Heverlee, Belgium.

Extending the results of [1], [2] and [3], it is shown that for a multiple-access channel with Slepian-Wolf correlated sources, it is possible to generate stepwise a code with only a few (linear in the blocklength n) permutations, keeping the average error probability upper bounded by an exponentially small quantity. In order to reach this result, special subgroups of S_n, the symmetric group of order n, are defined, whose elements, applied on the separate members of codeword pairs of a specified joint type, do not alter this joint type.

1. PRELIMINARIES

The concepts and results regarding types and information quantities used are by now standard and can be found in [1], [4] and [5]. However, some notations we use in the sequel, are recalled first.

For finite sets X, Y, ..., let $P(X)$ denote the set of probability distributions on X, $|X|$ the cardinality of X, $W(X,Y)$ the set of stochastic matrices with $|X|$ rows and $|Y|$ columns, and $P_n(X)$ the set of n-types (or types) on X^n, i.e. those elements of $P(X)$ which are possible frequency distributions of the letters of X in sequences $\underline{x} \in X^n$. For $p \in P_n(X)$, $W_n^p(X,Y)$ consists of those $w \in W(X,Y)$ for which p.w (defined by $p.w(x,y) \triangleq p(x)w(y|x)$) is a possible frequency distribution on the pairs of $X \times Y$ in sequences $(\underline{x},\underline{y}) \in X^n \times Y^n$. For $p \in P_n(X)$ define $T_p^n = \{\underline{x} \in X^n \mid \underline{x} \text{ has type } p\}$ and for $p \in P_n(X)$, $\underline{x} \in T_p^n$, $w \in W_n^p(X,Y)$, the set $T_w^n(\underline{x}) = \{\underline{y} \in Y^n \mid (\underline{x},\underline{y}) \text{ has joint type } p.w\}$ is called the w-shell of $\underline{x}$ and its elements are said to have conditional type w given $\underline{x}$.

2. THE CHANNEL MODEL AND A FIXED COMPOSITION CODE FOR IT

We consider the general situation introduced by Slepian and Wolf [6] of a discrete memoryless (d.m.) multiple-access channel (MAC) with two encoders, one decoder and three message sources as pictured in Fig. 1. M_1 and M_2, each observed by one of the encoders, are the private message sets and M_0, the set of common messages, is encoded by both transmitters. X and Y are the input alphabets, Z is the output alphabet and $w = w(z|xy)$ the transition probability matrix. We denote this communication situation by $SW(w : X \times Y \to Z; M_0, M_1, M_2)$.

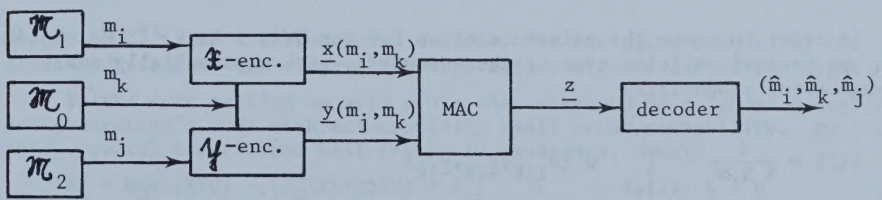


Fig. 1. Slepian-Wolf communication situation

★Supported by Project GOA 83/88-53, Ministerie Wetenschapsbeleid, Brussels.

In order to define a fixed composition code (FCC) for it, we first assume that an auxiliary encoder generates an auxiliary code $\{\underline{u}_k \mid k=1,\ldots,|M_0|\}$ for the common messages, the codewords of which are of fixed type $r \in P_n(U)$, where U is an auxiliary finite alphabet. Both the X- and the Y-encoder are supposed to know this auxiliary code and to condition their codewords on it as follows. Let $v_X \in W(U,X)$ and $v_Y \in W(U,Y)$ so that $r.v_X = p \in P_n(X)$ and $r.v_Y = q \in P_n(Y)$, and take $\underline{x}_{ik} \in T^n_{v_X}(\underline{u}_k)$ and $\underline{y}_{jk} \in T^n_{v_Y}(\underline{u}_k)$ for $i=1,\ldots,N_1$; $j=1,\ldots,N_2$; $k=1,\ldots,N_0$, whereby $N_s = |M_s|$, $s=0,1,2$. Now for each partition $(D_{ijk})_{i,j,k}$ of Z^n

$$C = \{(\underline{x}_{ik},\underline{y}_{jk},D_{ijk}) \mid i=1,\ldots,N_1;\ j=1,\ldots,N_2;\ k=1,\ldots,N_0\}$$

is an FCC for the given communication situation.

The X-codewords have fixed type p and the Y-codewords fixed type q. But this way the joint type of codeword pairs $(\underline{x}_{ik},\underline{y}_{jk})$ is not specified. This leads to difficulties when expressing the channel transition probabilities by means of the information-theoretical divergence between stochastic matrices, conditioned on the joint distribution of the inputs, as is usually done when upper bounding the error probabilities of FCC. In order to cope with this problem we will fix the joint type of codeword pairs by proceeding as follows. In Fig. 2. we define the canonical codeword triple of the given FCC C. All its codeword triples $(\underline{x}_{ik}, \underline{y}_{jk}, \underline{u}_k)$ must then have the same type as this canonical triple.

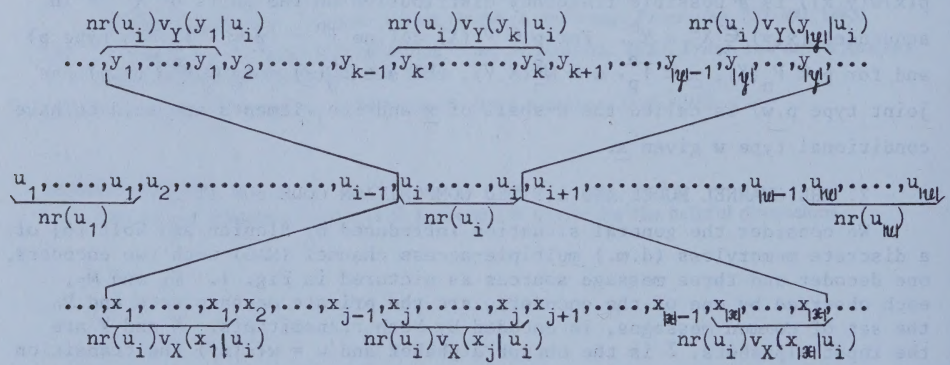


Fig. 2. Canonical codeword triple $(\underline{\hat{x}},\underline{\hat{u}},\underline{\hat{y}})$

In order to prove the existence of an FCC for $SW(w : X \times Y \to Z; M_0, M_1, M_2)$ with the prescribed joint type of the codewords, with exponentially small average error probability

$$\lambda(C) = \frac{1}{N_0 N_1 N_2} \sum_{i,j,k} w^n(D^c_{ijk} | \underline{x}_{ik},\underline{y}_{jk})$$

and generated by permutations, first suitable subgroups in S_n are constructed and then a codeproducer can be defined.

3. A CODEPRODUCER PRESERVING THE JOINT TYPE OF CODEWORD PAIRS

Let the canonical codeword triple $(\underline{\hat{x}},\underline{\hat{y}},\underline{\hat{u}})$ be given. Then define $f^i_{y_k} = nr(u_i)v_Y(y_k|u_i)$ and $f^i_k(\underline{\hat{y}}) = \sum_{j \leqslant k} f^i_{y_j}$, for $i=1,\ldots,|\mathcal{U}|$; $k=1,\ldots,|\mathcal{Y}|$; and analogously $f^i_{x_k}$ and $f^i_k(\underline{\hat{x}})$. Then put $f^i_k = \max(f^i_k(\underline{\hat{x}}), f^i_k(\underline{\hat{y}}))$, $i=1,\ldots,|\mathcal{U}|$; $k=1,\ldots,\min(|\mathcal{X}|,|\mathcal{Y}|)$.

Now subsets S_X and S_Y of S_n are constructively defined as follows :

- if $f^i_k = f^i_k(\underline{\hat{x}}) > f^i_k(\underline{\hat{y}})$, then add to S_Y the permutations that permute the letters of $\underline{\hat{y}}$ indexed from $f_i + f^i_{k-1} + 1$ to $f_i + f^i_k$, but leave the other letters fixed; hereby $f_i = \sum_{j<i} nr(u_j)$;
- if $f^i_k = f^i_k(\underline{\hat{y}}) > f^i_k(\underline{\hat{x}})$, then proceed analogously for S_X;
- if $f^i_k = f^i_k(\underline{\hat{x}}) = f^i_k(\underline{\hat{y}})$, then look if either $\underline{\hat{x}}$ or $\underline{\hat{y}}$ has different letters of its alphabet on the places indexed from $f_i + f^i_{k-1} + 1$ to $f_i + f^i_k$ and add to the corresponding set (either S_X or S_Y), the permutations which permute these different letters and leave the other ones fixed; if none of the two sequences $\underline{\hat{x}}$ or $\underline{\hat{y}}$ has different letters in the considered interval, then do nothing.

Now G_X is the subgroup of S_n generated by S_X and G_Y is defined similarly.

A codeproducer for $SW(w : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}; M_o, M_1, M_2)$ is a triple of subsets : $(\sigma_1,\ldots,\sigma_{nk_1}) \in G_X$, $(\tau_1,\ldots,\tau_{nk_o}) \in S_n$ and $(\gamma_1,\ldots,\gamma_{nk_2}) \in G_Y$, whereby $nk_s = \lceil nR_s \rceil$, $R_s = (\log N_s)/n$, $s = 0,1,2$. It works as follows. Let M_s be binarily represented in $\{0,1\}^{nk_s}$. If $m_i \in M_1$ is represented by $(s_1,\ldots,s_{nk_1})$ $m_k \in M_o$ by $(t_1,\ldots,t_{nk_o})$ and $m_j \in M_2$ by $(c_1,\ldots,c_{nk_2})$,

then (m_i,m_k) is X-encoded by $\tau_{nk_o}^{t_{nk_o}} \circ \ldots \circ \tau_1^{t_1} \circ \sigma_{nk_1}^{s_{nk_o}} \circ \ldots \circ \sigma_1^{s_1}\ (\underline{\hat{x}})$

and (m_j,m_k) is Y-encoded by $\tau_{nk_o}^{t_{nk_o}} \circ \ldots \circ \tau_1^{t_1} \circ \gamma_{nk_2}^{c_{nk_2}} \circ \ldots \circ \gamma_1^{c_1}\ (\underline{\hat{y}})$.

Denote by $C^{\sigma\tau\gamma}$ the code generated this way for the given channel by the given codeproducer and observe that its codeword pairs are of the right type.

In the next section we will prove the existence of a "good" codeproducer generating a code with exponentially small error probability. By doing this, several expressions will figure in exponents, namely :

$$B_X = H^{\approx}_w(Z|XYU) - |I^{\sim}_w(X;YZ|TU) - R_1|^+$$

$$B_Y = H^{\approx}_w(Z|XYU) - |I^{\sim}_w(Y;XZ|TU) - R_2|^+$$

$$B_{XY} = H_{\underset{\sim}{w}}^{\approx}(Z|XYU) - |I_{\underset{\sim}{w}}(XY;Z|TU) - (R_1 + R_2)|^+$$

$$B = H_{\underset{\sim}{w}}^{\approx}(Z|XYU) - |I_{\underset{\sim}{w}}(XYU;Z) - (R_o + R_1 + R_2)|^+$$

$$E(SW) = \min_{\widetilde{w}} D(\widetilde{w} \| w|\hat{p}) + A(\widetilde{w}),\ E_o(SW) = \min_{\widetilde{w}} D(\widetilde{w} \| w|\hat{p}) + A_o(\widetilde{w})$$

$$A(\widetilde{w}) = H_{\underset{\sim}{w}}^{\approx}(Z|XYU) - \max(B_X, B_Y, B_{XY}, B),\ A_o(\widetilde{w}) = H_{\underset{\sim}{w}}^{\approx}(Z|XYU) - \max(B_X, B_Y, B_{XY})$$

Hereby T is a random variable with values in $\mathcal{T}$, $|\mathcal{T}| = \min(|\mathcal{X}|, |\mathcal{Y}|)$ and with conditional distribution v_T given U, defined by

$$nv_T(t_k|u_i) = f_k^i - f_{k-1}^i .$$

The joint type $\hat{p}$ on (X,U,Y,T) is the one fixed by the quadruple $(\underline{\hat{x}},\underline{\hat{u}},\underline{\hat{y}},\underline{\hat{t}})$, whereby $\underline{\hat{t}}$ consists of $|\mathcal{U}|$ blocks, the i-th one starting with $v_T(t_1|u_i)$ times the letter t_1, followed by $v_T(t_2|u_i)$ repetitions of t_2 and ending with $v_T(t_{|\mathcal{T}|}|u_i)$ times the last letter $t_{|\mathcal{T}|}$ of $\mathcal{T}$. The necessity of the conditioning on T is a consequence of the joint type imposed on the codewords.

4. THE EXISTENCE OF A GOOD CODEPRODUCER

The proof of the existence of a "good" codeproducer consists of two parts :

1) the proof of the existence of a good codeproducer consisting of elements of G_X and G_Y, constructed with the sole purpose of preserving a specified joint type on codeword pairs for a classical MAC, i.e., without a common source (Theorem 1);
2) the proof of the fact that, once one disposes of a good code for the classical MAC, it is possible to multiply this code progressively, by means of elements in S_n, in order to obtain a good code for SW(w : $\mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$; M_0, M_1, M_2) (Theorem 2).

Theorem 1 :

Let $C = \{(\underline{x}_i, \underline{y}_j, D_{ij}) \mid i=1,\ldots,N_1;\ j=1,\ldots,N_2\}$ be an FCC for MAC(w : $\mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$; M_1, M_2), with $\underline{x}_i \in T^n_{v_X}(\underline{\hat{u}})$, $\underline{y}_j \in T^n_{v_Y}(\underline{\hat{u}})$, such that the joint type of $(\underline{x}_i, \underline{y}_j, \underline{\hat{u}})$ is that of $(\underline{\hat{x}}, \underline{\hat{y}}, \underline{\hat{u}})$. Then there exist $\sigma \in G_X$, $\gamma \in G_Y$ and decoding sets $D'_{ij} \subset D_{ij}$; $D_{\sigma i,j}$; $D_{i,\gamma j}$; $D_{\sigma i,\gamma j}$; such that the code

$$C^{\sigma\gamma} = \{(\underline{x}_i, \underline{y}_j, D'_{ij}), (\sigma\underline{x}_i, \underline{y}_j, D_{\sigma i,j}), (\underline{x}_i, \gamma\underline{y}_j, D_{i,\gamma j}), (\sigma\underline{x}_i, \gamma\underline{y}_j, D_{\sigma i,\gamma j}) \mid i=j,\ldots,N_1;\ j=1,\ldots,N_2\}$$

for MAC(w : $\mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$; M'_1, M'_2), $|M'_1| = 2N_1$, $|M'_2| = 2N_2$, has its average error probability upper bounded in the following way :

$$\lambda(C^{\sigma\gamma}) \leq \lambda(C) + 2N_1N_2 \exp\{-n(E_o(SW) - \delta)\},$$

for all $\delta > 0$ provided n is sufficiently large.

Proof :

A. Permutations σ and γ are chosen such that (σ,γ) is a realisation of the uniform random variable (Σ,Γ) on $G_X \times G_Y$, satisfying for all $\widetilde{w}, \overset{\approx}{w}$:

$$\max(\hat{g}_X(\sigma^{-1}),\hat{g}_X(\sigma)) \leqslant N_1 \exp\{n(B_X(\tilde{w},\tilde{\tilde{w}}) + \frac{\delta}{2}\}$$

$$\max(\hat{g}_Y(\gamma^{-1}),g_Y(\gamma)) \leqslant N_2 \exp\{n(B_Y(\tilde{w},\tilde{\tilde{w}}) + \frac{\delta}{2}\}$$

$$\max(\hat{g}_{XY}(\sigma^{-1},\gamma^{-1}),\hat{g}_Y(\sigma,\gamma)) \leqslant N_1N_2 \exp\{n(B_{XY}(\tilde{w},\tilde{\tilde{w}}) + \frac{\delta}{2}\}$$

whereby

$$\hat{g}_X(\sigma) \triangleq \sum_{i=1}^{N_1} g_X(\sigma\underline{x}_i); \hat{g}_Y(\gamma) \triangleq \sum_{j=1}^{N_2} g_Y(\gamma\underline{y}_j); \ \hat{g}_{XY}(\sigma,\gamma) \triangleq \sum_{i,j} g_{XY}(\sigma\underline{x}_i,\gamma\underline{y}_j);$$

$$g_X(\underline{x}) \triangleq \frac{1}{N_2} \sum_{j=1}^{N_2} |T^n_{\tilde{w}}(\underline{x},\underline{y}_j,\hat{\underline{u}}) \cap \bigcup_{i=1}^{N_1} T^n_{\tilde{\tilde{w}}}(\underline{x}_i,\underline{y}_j,\hat{\underline{u}})|,$$

$$g_Y(\underline{y}) \triangleq \frac{1}{N_1} \sum_{i=1}^{N_1} |T^n_{\tilde{w}}(\underline{x}_i,\underline{y},\hat{\underline{u}}) \cap \bigcup_{j=1}^{N_2} T^n_{\tilde{\tilde{w}}}(\underline{x}_i,\underline{y}_j,\hat{\underline{u}})|,$$

$$g_{XY}(\underline{x},\underline{y}) \triangleq |T^n_{\tilde{w}}(\underline{x},\underline{y},\hat{\underline{u}}) \cap \bigcup_{i=1}^{N_1} \bigcup_{j=1}^{N_2} T^n_{\tilde{\tilde{w}}}(\underline{x}_i,\underline{y}_j,\hat{\underline{u}})|,$$

for $\underline{x} \in G_X(\hat{\underline{x}}) \triangleq \{\sigma\hat{\underline{x}} \mid \sigma \in G_X\}$ and $\underline{y} \in G_Y(\hat{\underline{y}}) \triangleq \{\gamma\hat{\underline{y}} \mid \gamma \in G_Y\}$.
Lemmata 1 and 3 (below) guarantee the existence of suitable σ and γ.

B. The new decoding sets are defined by the following construction. Let

$$\hat{D}_{i,\gamma j} = (\bigcup_j D_{ij}) \cap \gamma D_{ij} \setminus \bigcup_k \{\underline{z} \mid I(\underline{y}_k;\underline{z}|\underline{x}_i\hat{\underline{u}}) \geqslant I(\gamma\underline{y}_j;\underline{z}|\underline{x}_i\hat{\underline{u}})\},$$

$$\hat{D}_{\sigma i,j} = (\bigcup_i D_{ij}) \cap \sigma D_{ij} \setminus \bigcup_k \{\underline{z} \mid I(\underline{x}_k;\underline{z}|\underline{u}_j\hat{\underline{u}}) \geqslant I(\sigma\underline{x}_i;\underline{z}|\underline{y}_j\hat{\underline{u}})\} \text{ and}$$

$\hat{D}_{ij} = D_{ij} \setminus ((\bigcup_j \hat{D}_{i,\gamma j}) \cup (\bigcup_i \hat{D}_{\sigma i,j}))$. Then we define

$$D_{\sigma i,\gamma j} \triangleq (\bigcup_i \hat{D}_{i,\gamma j} \cap \bigcup_j \hat{D}_{\sigma i,j}) \cup (\gamma\sigma D_{ij} \setminus I_{\sigma\underline{x}_i,\gamma\underline{y}_j})$$

with $I_{\sigma\underline{x}_i,\gamma\underline{y}_j} \triangleq \bigcup_{k,\ell} \{\underline{z} \mid I(\underline{x}_k\underline{y}_\ell;\underline{z}|\hat{\underline{u}}) \geqslant I(\sigma\underline{x}_i\gamma\underline{y}_j;\underline{z}|\hat{\underline{u}})$, and

$$D_{i,\gamma j} \triangleq \hat{D}_{i,\gamma j} \setminus \bigcup_i D_{\sigma i,\gamma j}; \ D_{\sigma i,j} \triangleq \hat{D}_{\sigma i,j} \setminus \bigcup_j D_{\sigma i,\gamma j}; \ D'_{ij} \triangleq \hat{D}_{ij} \setminus \bigcup_{i,j} D_{\sigma i,\gamma j}.$$

C. Finally one calculates the error probability of the new code.

$$4N_1N_2\lambda(C^{\sigma\gamma})$$

$$= \sum_i \sum_j (w^n(D'^c_{ij}|\underline{x}_i,\underline{y}_j) + w^n(D^c_{\sigma i,j}|\sigma\underline{x}_i,\underline{y}_j) + w^n(D^c_{i,\gamma j}|\underline{x}_i,\gamma\underline{y}_j)$$
$$+ w^n(D^c_{\sigma i,\gamma j}|\sigma\underline{x}_i,\gamma\underline{y}_j))$$

$$= N_1N_2\lambda(C) + \sum_i \sum_j (w^n(D_{ij} \setminus D'_{ij}|\underline{x}_i,\underline{y}_j) + w^n(D^c_{\sigma i,j}|\sigma\underline{x}_i,\underline{y}_j)$$
$$+ w^n(D^c_{i,\gamma j} \mid \underline{x}_i,\gamma\underline{y}_j) + w^n(D^c_{\sigma i,\gamma j}|\sigma\underline{x}_i,\gamma\underline{y}_j)).$$

Now we have

$$(a)\quad \bigcup_{i,j}(D_{ij}\setminus D'_{ij}) \subset \bigcup_{i,j}\Big(\big(\bigcup_{j}\hat{D}_{i,\gamma j}\big)\cup\big(\bigcup_{i}\hat{D}_{\sigma i,j}\big)\cup\big(\bigcup_{i,j}D_{\sigma i,\gamma j}\big)\Big)$$

$$\subset \bigcup_{i,j}\Big(\bigcup_{k}\{\underline{z}\mid I(\gamma\underline{y}_k;\underline{z}|\underline{x}_i\underline{u}) > I(\underline{y}_j;\underline{z}|\underline{x}_i\hat{\underline{u}})\}\cup$$

$$\cup\bigcup_{k}\{\underline{z}\mid I(\sigma\underline{x}_k;\underline{z}|\underline{y}_j\hat{\underline{u}}) > I(\underline{x}_i;\underline{z}|\underline{y}_j\hat{\underline{u}})\}\cup$$

$$\cup\bigcup_{k,\ell}\{\underline{z}\mid I(\sigma\underline{x}_k\gamma\underline{y}_\ell;\underline{z}|\hat{\underline{u}}) > I(\underline{x}_i\underline{y}_j;\underline{z}|\hat{\underline{u}})\}\Big);$$

$$(b)\quad \bigcup_{i,j}(D^c_{i,\gamma j}) \subset \bigcup_{i,j}(\gamma D^c_{ij}\cup\bigcup_{k}\{\underline{z}\mid I(\underline{y}_k;\underline{z}|\underline{x}_i\hat{\underline{u}})\geqslant I(\gamma\underline{y}_j;\underline{z}|\underline{x}_i\hat{\underline{u}})\}\cup I_{\sigma\underline{x}_i,\gamma\underline{y}_j});$$

$$(c)\quad \bigcup_{i,j}(D^c_{\sigma i,j}) \subset \bigcup_{i,j}(\sigma D^c_{ij}\cup\bigcup_{k}\{\underline{z}\mid I(\underline{x}_k;\underline{z}|\underline{y}_j\hat{\underline{u}})\geqslant I(\sigma\underline{x}_i;\underline{z}|\underline{y}_j\hat{\underline{u}})\}\cup I_{\sigma\underline{x}_i,\gamma\underline{y}_j});$$

$$(d)\quad \bigcup_{i,j}(D^c_{\sigma i,\gamma j}) \subset \bigcup_{i,j}(\gamma\sigma D^c_{ij}\cup I_{\sigma\underline{x}_i,\gamma\underline{y}_j}).$$

Now using the requirements imposed on σ and γ in A. and the inclusions (a), (b), (c) and (d), one obtains,

$$4N_1N_2\lambda(C^{\sigma\gamma}) \leqslant 4N_1N_2\lambda(C) + 8N_1N_2\exp\{-n(E_o(SW)-\delta)\}$$

as was to be proven.

Further observe that if $\lambda(C) \leqslant \exp\{-n(E_o(SW)-\delta)\}$, then also

$$\lambda(C^{\sigma\gamma}) \leqslant 3\exp\{-n(E_o(SW)-\delta)\}.$$

Theorem 2 :

Let $C = \{(\underline{x}_{ik},\underline{y}_{jk},D_{ijk}) \mid k=1,\ldots,N_o;\ i=1,\ldots,N_1;\ j=1,\ldots,N_2\}$ be an FCC with auxiliary code $(\underline{u}_k)_k$ for $SW(w : X\times Y\to Z;\ M_o,\ M_1,\ M_2)$, as defined above. Then there exist $\tau\in S_n$; $D'_{ijk}\subset D_{ijk}$; $D_{\tau ijk}\subset\tau D_{ijk}$; such that

$$C^\tau = \{(\underline{x}_{ik},\underline{y}_{jk},D'_{ijk}),(\tau\underline{x}_{ik},\tau\underline{y}_{jk},D_{\tau ijk}) \mid i,j,k\},$$

is an FCC for $SW(w : X\times Y\to Z;\ M'_o,\ M_1,\ M_2)$ with $|M'_o| = 2N_o$, for which

$$2N_oN_1N_2\lambda(C^\tau) \leqslant N_oN_1N_2\lambda(C) + N_oN_1N_2\exp\{-n(E(SW)-\delta)\},$$

for all positive δ and sufficiently large n.

Proof :

A. The permutation τ is chosen to be the realisation of an uniform random variable T on S_n, satisfying

$$\max(\hat{g}(\tau),\hat{g}(\tau^{-1})) \leqslant N_o\exp\{n(B(\tilde{w},\tilde{\tilde{w}}) + \tfrac{\delta}{2})\},$$

where, for all τ : $\hat{g}(\tau) \triangleq \sum_{k=1}^{N_o} g((\tau\underline{x}_{ik})_i,(\tau\underline{y}_{jk})_j,\tau\underline{u}_k)$

and for all $(\underline{x}_i)_i \in (G_X(\hat{\underline{x}}))^{N_1}$, $(\underline{y}_j)_j \in (G_Y(\hat{\underline{y}}))^{N_2}$, $\underline{u} \in T_r^n$:

$$g((\underline{x}_i)_i,(\underline{y}_j)_j,\underline{u}) \triangleq \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \left| T_w^n(\underline{x}_i,\underline{y}_j,\underline{u}) \cap \bigcup_{k,\ell,m} T_w^n(\underline{x}_{\ell k},\underline{y}_{mk},\underline{u}_k) \right| .$$

Because of Lemmata 2 and 4 below we know that such a τ exists.

B. New decoding sets are defined as follows :

$$D'_{ijk} \triangleq D_{ijk} \setminus \bigcup_{\ell mn} \{\underline{z} \mid I(\underline{z};\tau\underline{x}_{\ell n}\tau\underline{y}_{mn}\tau\underline{u}_n) \geq I(\underline{z};\underline{x}_{jk}\underline{y}_{ik}\underline{u}_k)\},$$

$$D'_{\tau ijk} \triangleq \tau D_{ijk} \setminus \bigcup_{\ell mn} \{\underline{z} \mid I(\underline{z};\underline{x}_{\ell n}\underline{y}_{mn}\underline{u}_n) > I(\underline{z};\tau\underline{x}_{jk}\tau\underline{y}_{ik}\tau\underline{u}_k) .$$

C. The calculation of the error probability goes as usual (cf. AMAC [3] and ABC [2]). This concludes this proof.

In order to conclude this section we observe that these two theorems effectively guarantee the existence of a good code $C^{\sigma\gamma\tau}$ for $SW(w : X \times Y \to Z; M_o, M_1, M_2)$. Starting with a code with one codeword pair $(\hat{\underline{x}},\hat{\underline{y}})$ and the auxiliary codeword $\hat{\underline{u}}$, one obtains a good code for the private messages by applying Theorem 1 $\lceil n(R_1+R_2) \rceil$ times. This code is then multiplied for the common messages by $\lceil nR_o \rceil$ applications of Theorem 2, thus resulting in a code for the given channel with average error probability not larger than $\exp\{-n(E(SW)-\delta)\}$, for all positive δ, provided n is sufficiently large.

5. CONCLUDING REMARKS

In the above, we have shown the possibility of generating by means of permutations a code for the SWMAC, having an exponentially small average error probability. However, the exponent in the error bound, resulting from our construction, is somewhat weaker than the one obtained by Slepian and Wolf [6], because of the conditioning on the random variable T. The latter is due to the restriction on the joint type of pairs of codewords. Indeed, the code construction proposed here has the feature to keep the joint type of codeword pairs fixed, thus facilitating the decoding and the calculation of the error probability, and to make a stepwise approach possible.

Finally, we like to point out that, as for the one-way channel in [1], the ABC in [2] and the AMAC in [3], the generation of codes by permutations implies an important gain from the point of view of storage. Indeed, only $n(R_0+R_1+R_2)$ permutations have to be stored in order to produce the two sets of codewords, one with $\exp\{n(R_0+R_1)\}$ and the other with $\exp\{n(R_0+R_2)\}$ elements. Of course, one could hope for still a greater gain in storage room, by imagining more algebraically structured sets of permutations for producing a code. However, one should keep in mind that then it could become more difficult to keep the error probability properly bounded.

ACKNOWLEDGEMENT

The author likes to thank her advisor Prof. dr. E.C. van der Meulen for the many discussions on the subject, which helped to improve this paper.

REFERENCES

[1] R. Ahlswede and G. Dueck, "Good codes can be produced by a few permutations", IEEE Trans. Inform. Theory, vol. IT-28, no. 3, pp. 430-443, May 1982.

[2] K. De Bruyn, "Codes for the asymmetric broadcast channel, generated by permutations", accepted for publication in IEEE Trans. Inform. Theory.

[3] K. De Bruyn and E.C. van der Meulen, "An iterative code construction and codes generated by permutations, for the asymmetric multiple-access channel", presented at IEEE-ISIT, St. Jovite, Canada, Sept. 1983, submitted for publication.

[4] I. Csiszár and J. Körner, Information Theory : Coding Theorems for Discrete Memoryless Systems, Akadémiai Kiado, Budapest, 1981.

[5] K. De Bruyn, forthcoming Ph. Dissertation, K.U.Leuven, 1984.

[6] D. Slepian and J.K. Wolf, "A coding theorem for multiple-access channels with correlated sources", Bell System Tech. J., vol. 52, no. 7, pp. 1037-1076, Sept. 1973.

APPENDIX

For all positive δ and sufficiently large n :

Lemma 1 : $E[g_X(X)] \leqslant \exp\{n(B_X(\tilde{\tilde{w}},\tilde{\tilde{\tilde{w}}}) + \frac{\delta}{2})\}$,

$E[g_Y(Y)] \leqslant \exp\{n(B_Y(\tilde{\tilde{w}},\tilde{\tilde{\tilde{w}}}) + \frac{\delta}{2})\}$,

$E[g_{XY}(X,Y)] \leqslant \exp\{n(B_{XY}(\tilde{\tilde{w}},\tilde{\tilde{\tilde{w}}}) + \frac{\delta}{2})\}$,

with (X,Y) uniform on $G_X(\underline{\hat{x}}) \times G_Y(\underline{\hat{y}})$, and with B_X, B_Y and B_{XY} defined in Section 3.

Lemma 2 : $E[g((X_i)_i, (Y_j)_j, U] \leqslant \exp\{n(B(\tilde{\tilde{w}},\tilde{\tilde{\tilde{w}}}) + \frac{\delta}{2})\}$,

with $(X_i,Y_j,U)_{i,j}$, i.i.d. according to the prescribed joint type.

Lemma 3 : $E[\hat{g}_X(\Sigma)] = N_1 E[g_X(X)]$,

$E[\hat{g}_Y(\Gamma)] = N_2 E[g_Y(Y)]$,

$E[\hat{g}_{XY}(\Sigma,\Gamma)] = N_1 N_2 E[g_{XY}(X,Y)]$.

Lemma 4 : $E[\hat{g}(T)] = N_o E[g((X_i)_i, (Y_j)_j, U)]$.

The proofs of these lemmata, consisting in calculating and upper bounding the expectations of the given functions, are rather technical, and especially involve the upper bounding of the cardinalities of orbits $G_X(\underline{\hat{x}})$, $G_Y(\underline{\hat{y}})$, of $\tilde{\tilde{w}}$- and $\tilde{\tilde{\tilde{w}}}$-shells, and of the probabilities that outputs $\underline{z}$ belong to such shells.

★

★ ★

TWO-DIMENSIONAL BLOCK INTERFERENCE CHANNELS WITH APPLICATIONS TO RANDOM ACCESS MEMORIES

WAYNE ERIC STARK
JAE HONG LEE
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109

ABSTRACT

We define channels with two-dimensional block interference (BI), and calculate their channel capacities in the presence or the absence of side information. We apply this channel model to a random access memory (RAM) with defective rows and columns, and compute its capacity. We show that the capacity, in bits per cell, is independent of the dimensions of the RAM when side information is available.

I INTRODUCTION

In this paper we consider a channel model that exhibits memory in two directions. We use this channel to model a random access memory. The RAM is assumed to be a array of cells. Errors occur in the storage and retrieval of information from the RAM for a variety of reasons. Certain of these errors may cause whole rows or whole columns of cells to be received as all 0's or all 1's (stuck-at faults) independent of what was written into the cell. Other error mechanisms can cause all bits within a row or column to be in error with large probability. These channels are clearly not memoryless from cell to cell. Several error-correction algorithms have been discussed in recent literature [5-7]. In this paper we determine the ultimate limits of coding by computing the channel capacity.

In Section 2, we define the channel model with two dimensional BI. In Section 3, we discuss the relationship between the number of rows and columns of a RAM and its capacity. In Section 4, we give sample calculations of the capacity of a semiconductor RAM.

II CHANNEL MODELS

We extend the model of channels with BI [1] to a model of channels which we call channels with two dimensional BI. Channels with two dimensional BI are characterized by random vectors $\underline{S}^{(r)}$ and $\underline{S}^{(c)}$ and two positive numbers L and M. The random vectors $\underline{S}^{(r)}$ and $\underline{S}^{(c)}$ describe row and column noise severities, respectively; the integers L and M measure the length of a given level of noise severity for each dimension. The presence or the absence of the information concerning the noise severity at the receiver produces four possible types of channels called as follows: the two-dimensional BI channel without side information, the two-dimensional BI channel with row side information, the two-dimensional BI channel with column side information, and the two-dimensional BI channel with row and column side information.

We now describe the four types of the two-dimensional BI channel shown in Figure 1. We start with a finite set of memoryless channels $\{\Delta_s; s \in \Omega\}$, each with input alphabet A and output alphabet B and capacity C_s. Let us assume that the index parameter s lies in a set Ω. Let us define $\underline{S}^{(r)}=(S_1^{(r)},S_2^{(r)},\ldots,S_L^{(r)})$ to be a sequence of independent identically distributed Ω-valued random variables with common distribution $P^{(r)}$. Also define $\underline{S}^{(c)}=(S_1^{(c)},S_2^{(c)},\ldots,S_M^{(c)})$ to be a sequence of independent identically distributed Ω-valued random variables with common distribution $P^{(c)}$. For each $L\geq 1$ and $M\geq 1$, define $\Delta_{S_1^{(r)},S_1^{(c)}},\ldots,\Delta_{S_1^{(r)},S_M^{(c)}},\ldots,\Delta_{S_L^{(r)},S_1^{(c)}},\ldots,\Delta_{S_L^{(r)},S_M^{(c)}}$ to be a sequence of channels determined by the row and the column state information vectors $\underline{S}^{(r)}$ and $\underline{S}^{(c)}$ with each channel $\Delta_{S_i^{(r)},S_j^{(c)}}$ being a channel from the collection of

channels $\{\Delta_s ; s \in \Omega\}$.

We now define the two-dimensional BI channel without side information. Consider a sequence of channel input symbols $\underline{X} = (X_{1,1}, \ldots, X_{1,M}, \ldots, X_{L,1}, \ldots, X_{L,M})$ from alphabet A, and a corresponding sequence of output symbols $\underline{Y}$ from alphabet B. Then the $((i-1)M+j)$-th symbol is transmitted over one of the component channels $\{\Delta_s\}$ which is determined by $S_i^{(r)}$ and $S_j^{(c)}$ in the following manner. If the capacities of $\Delta_{S_i^{(r)}}$ and $\Delta_{S_j^{(c)}}$ are unequal, then $\Delta_{S_i^{(r)}, S_j^{(c)}} = min\ \{\Delta_{S_i^{(r)}}, \Delta_{S_j^{(c)}}\}$, where the '*min*' of two channels is one with the smaller capacity. If the capacities are equal, either one is chosen with equal probability. In this manner, the state of a cell marked by '!' in Figure 1 is determined. If the receiver has no knowledge of $\underline{S}^{(r)}, \underline{S}^{(c)}$, then we say the channel does not have side information and is denoted by $\Delta(L,M)$. If the receiver has knowledge of $\underline{S}^{(r)}$, the channel is called the channel with row side information and denoted by $\Delta_{\underline{S}^{(r)}}(L,M)$. If the receiver has knowledge of $\underline{S}^{(c)}$, the channel is called the channel with column side information and denoted by $\Delta_{\underline{S}^{(c)}}(L,M)$. If the receiver has knowledge of both of $\underline{S}^{(r)}$ and $\underline{S}^{(c)}$, the channel is called the channel with row and column side information and denoted by $\Delta_{\underline{S}^{(r)}, \underline{S}^{(c)}}(L,M)$.

So far we have only described the channel behavior for a sequence of LM letters from the finite alphabet A. We imagine the next sequence of LM letters to be stored on another semiconductor chip which has an identical (but independent) distribution for the row and column noise severities $\underline{S}^{(r)}$ and $\underline{S}^{(c)}$. Similarly the next sequence of LM inputs is stored on another chip with independent and identically distributed noise. In this way the channel without side information is a discrete memoryless channel with input alphabet A^{LM} and output alphabet B^{LM}. Because of this the maximum rate at which information can be transmitted (stored) is given by the maximum of the mutual information between the channel input $\underline{X}$ and output $\underline{Y}$. Similar statements can be made for the channels with row and/or column side information.

III CAPACITY OF CHANNELS WITH TWO DIMENSIONAL BLOCK INTERFERENCE

In all our proofs in this section, we will make the following compatibility assumption about the component channels $\{\Delta_s\}$:

> Input distribution which achieves capacity is the same
> for each of the component channels Δ_s.

Let us denote the capacities of $\Delta(L,M)$, $\Delta_{\underline{S}^{(r)}}(L,M)$, $\Delta_{\underline{S}^{(c)}}(L,M)$, and $\Delta_{\underline{S}^{(r)}, \underline{S}^{(c)}}(L,M)$, per cell, by $C(L, M)$, $C_r(L, M)$, $C_c(L, M)$, and $C_{r,c}(L, M)$ respectively. For $\Delta(L,M)$,

$$C(L, M) = \frac{1}{LM} \max_{\underline{X}} I(\underline{X} ; \underline{Y}) . \tag{3.1}$$

For $\Delta_{\underline{S}^{(r)}}(L,M)$,

$$C_r(L, M) = \frac{1}{LM} \max_{\underline{X}} I(\underline{X} ; \underline{Y}, \underline{S}^{(r)}) . \tag{3.2}$$

For $\Delta_{\underline{S}^{(c)}}(L,M)$,

$$C_c(L, M) = \frac{1}{LM} \max_{\underline{X}} I(\underline{X} ; \underline{Y}, \underline{S}^{(c)}) . \tag{3.3}$$

For $\Delta_{\underline{S}^{(r)}, \underline{S}^{(c)}}(L,M)$,

$$C_{r,c}(L, M) = \frac{1}{LM} \max_{\underline{X}} I(\underline{X} ; \underline{Y}, \underline{S}^{(r)}, \underline{S}^{(c)}) . \tag{3.4}$$

Since $I(\underline{X} ; \underline{Y}) \leq I(\underline{X} ; \underline{Y}, \underline{S}^{(r)})$ [2, Theorem 1.4], it follows immediately that for all L and M

$$C(L, M) \leq C_r(L, M) \leq C_{r,c}(L, M) \tag{3.5}$$

$$C(L, M) \leq C_c(L, M) \leq C_{r,c}(L, M) \tag{3.6}$$

Let us consider the capacity of $\Delta_{\underline{S}^{(r)}, \underline{S}^{(c)}}(L,M)$. Since $I(\underline{X} ; \underline{S}^{(r)}, \underline{S}^{(c)}) = 0$,

$$I(\underline{X}; \underline{Y}, \underline{S}^{(r)}, \underline{S}^{(c)}) = I(\underline{X}; \underline{Y} | \underline{S}^{(r)}, \underline{S}^{(c)})$$

$$= E[I(\underline{X}; \underline{Y}^{(\underline{S}^{(r)}, \underline{S}^{(c)})})] \tag{3.7}$$

where $\underline{Y}^{(\underline{S}^{(r)}, \underline{S}^{(c)})}$ is the output vector when $\underline{X}$ is the input vector of the two-dimensional BI channel with row and column state vectors $\underline{S}^{(r)}$ and $\underline{S}^{(c)}$ and the expectation is with respect to the random vectors $\underline{S}^{(r)}$ and $\underline{S}^{(c)}$. Since

$$p(\underline{y} | \underline{x}, \underline{s}^{(r)}, \underline{s}^{(c)}) = \prod_{i=1}^{L} \prod_{j=1}^{M} p(y_{i,j} | x_{i,j}, s_i^{(r)}, s_j^{(c)}),$$

We have

$$I(\underline{X}; \underline{Y}, \underline{S}^{(r)}, \underline{S}^{(c)}) \leq LM\, E[I(X; Y^{(S)})] \tag{3.8}$$

with equality if and only if the input vector $\underline{X}$ has independent components [2. Theorem 1.9]. Also $Y^{(s)}$ is the output when X is the input of component channel Δ_s and the expectation is with respect to the random variable S whose distribution P is determined by $P^{(r)}$ and $P^{(c)}$ as follows. Let $S^{(r)}$ and $S^{(c)}$ be random variables with distribution $P^{(r)}$ and $P^{(c)}$ respectively. Then

$$P\{S = k\} = P\{S^{(r)} = k\}\, P\{S^{(c)} = k\}$$

$$+ P\{S^{(r)} = k\}\, P\{S^{(c)} \in \{j : C_j \succ C_k\}\}$$

$$+ P\{S^{(c)} = k\}\, P\{S^{(r)} \in \{j : C_j \succ C_k\}\}$$

which is independent of L and M. From (3.4) and (3.8), and by the compatibility assumption,

$$C_{r,c}(L, M) = E[C_S] \tag{3.9}$$

$$\equiv \overline{C}$$

which is independent of L and M.

Let us consider the capacity, $C_r(L, M)$, of the channel with row side information $\Delta_{\underline{S}^{(r)}}(L,M)$. Since $I(\underline{X}; \underline{S}^{(r)}) = 0$,

$$I(\underline{X}; \underline{Y}, \underline{S}^{(r)}) = I(\underline{X}; \underline{Y} | \underline{S}^{(r)})$$

$$= E[I(\underline{X}; \underline{Y}^{(\underline{S}^{(r)})})] \tag{3.10}$$

where $\underline{Y}^{(\underline{S}^{(r)})}$ is the output vector when $\underline{X}$ is the input vector of the two-dimensional BI channel with row state vector $\underline{S}^{(r)}$ and the expectation is with respect to the random vector $\underline{S}^{(r)}$. Note that

$$p(\underline{y} | \underline{x}, \underline{s}^{(r)}) = \prod_{j=1}^{M} p(\underline{\hat{y}}_j | \underline{\hat{x}}_j, \underline{s}^{(r)})$$

where $\underline{\hat{x}}_j = (x_{1,j}, x_{2,j}, \ldots, x_{L,j})$ and $\underline{\hat{y}}_j = (y_{1,j}, y_{2,j}, \ldots, y_{L,j})$. Since for each $\underline{s}^{(r)}$ the channel is memoryless from column to column,

$$I(\underline{X}; \underline{Y}, \underline{S}^{(r)}) \leq M\, E[I(\underline{\hat{X}}; \underline{\hat{Y}}^{(\underline{S}^{(r)})})] \tag{3.11}$$

with equality if and only if $\underline{\hat{x}}_1, \ldots, \underline{\hat{x}}_M$ are independent [2. Theorem 1.9]. Also $\underline{\hat{Y}}^{(\underline{s}^{(r)})}$ is the output vector when $\underline{X}$ is the input vector of any column of the two-dimensional BI channel with row side information $\underline{S}^{(r)}$. In (3.11), $E[I(\underline{\hat{X}}; \underline{\hat{Y}}^{(\underline{S}^{(r)})})]$ is independent of M and the expectation is with respect to the random vector $\underline{S}^{(r)}$. From (3.2) and (3.11), and by the compatibility assumption,

$$C_r(L, M) = \frac{1}{L}\, E[\hat{C}_{\underline{S}^{(r)}}] \tag{3.12}$$

where $\hat{C}_{\underline{s}^{(r)}}$ is the capacity of a column of the two-dimensional channel with row side information $\underline{s}^{(r)}$. Thus $C_r(L, M)$ is independent of M. Note that the input distribution which achieves capacity is not necessarily cell-wise independently distributed but is independent and identically

distributed from column to column. One can use Arimoto/Blahut's algorithm [3,4] to find optimal input distribution of the input vector $\underline{X}$. Also note that we have reduced complexity of calculation by requiring optimization of capacity over input vector $\underline{X}$ of length L instead of $\underline{X}$ of length LM. Similarly, for the channel with column side information $\underline{\Delta}_{\underline{S}^{(c)}}(L,M)$ we get

$$C_c(L, M) = \frac{1}{M} \mathrm{E}\left[\widetilde{C}_{\underline{S}^{(c)}}\right] \tag{3.13}$$

where $\widetilde{C}_{\underline{S}^{(c)}}$ is the capacity of a row of the two-dimensional channel with column side information $\underline{S}^{(c)}$. Thus $C_c(L, M)$ is independent of L.

Now we claim that

$$\lim_{L \to \infty} C_r(L, M) = \overline{C} \tag{3.14}$$

To prove (3.14), we require a lemma about average mutual information whose proof is omitted.

Lemma

$$I(X ; Y, Z) \leq I(X ; Y) + I(X, Y ; Z) \tag{3.15}$$

with equality if and only if Y and Z are independent.

Using (3.15), we have

$$I(\underline{X} ; \underline{Y}, \underline{S}^{(r)}) \leq I(\underline{X} ; \underline{Y}, \underline{S}^{(r)}, \underline{S}^{(c)}) \leq I(\underline{X} ; \underline{Y}, \underline{S}^{(r)}) + I(\underline{X}, \underline{Y}, \underline{S}^{(r)} ; \underline{S}^{(c)}) . \tag{3.16}$$

Our claim (3.14) will follow from the definitions (3.2) and (3.4), provided that we can show that

$$\lim_{L \to \infty} \frac{1}{LM} I(\underline{X}, \underline{Y}, \underline{S}^{(r)} ; \underline{S}^{(c)}) = 0 . \tag{3.17}$$

This was shown in [1] when the input and the output alphabets A and B are finite. A similar calculation shows that

$$\lim_{M \to \infty} C_c(L, M) = \overline{C} . \tag{3.18}$$

Finally we claim that

$$\lim_{L \to \infty} C(L, M) = C_c(L, M) . \tag{3.19}$$

Using the lemma (3.15), we have

$$I(\underline{X} ; \underline{Y}, \underline{S}^{(c)}) \leq I(\underline{X} ; \underline{Y}) + I(\underline{X}, \underline{Y} ; \underline{S}^{(c)}) . \tag{3.20}$$

Our claim (3.19) will follow from the compatibility assumption and (3.3), provided that we can show that

$$\lim_{L \to \infty} \frac{1}{LM} I(\underline{X}, \underline{Y} ; \underline{S}^{(c)}) = 0 . \tag{3.21}$$

The proof of (3.21) is essentially the same as that of (3.17). Similarly, we get

$$\lim_{M \to \infty} C(L, M) = C_r(L, M) . \tag{3.22}$$

From (3.14), (3.18), (3.19) and (3.22) we obtain

$$\lim_{M \to \infty} \lim_{L \to \infty} C(L, M) = \lim_{L \to \infty} \lim_{M \to \infty} C(L, M) = \overline{C} \tag{3.23}$$

The capacity of the channel with row and column side information, in bits per cell, does not depend on the dimensions L and M. The capacity of the channel with row side information does not depend on the number of columns; and the capacity of the channel with column side information does not depend on the number of rows. Also the capacity of the channel with row side information is less than but approaches the capacity of channel with row and column side information, as the the number of rows and columns increases. Similarly for the capacity of the

channel with column side information. In the next section we give some examples of two-dimensional BI channels.

IV EXAMPLES

In this section we consider three examples of two-dimensional BI channels that model defective RAMs.

Example 1 (A RAM with row and column defects)

We model a RAM with defective rows and columns as a channel with two dimensional BI. If a row is defective then every cell in that row is defective. Similarly for columns. Let us assume that a RAM consists of L rows and M columns of cells; that $\Omega=\{0,1,2\}$; that Δ_0 is a binary symmetric channel (BSC) with crossover probability p_o, Δ_1 is a "stuck-at-0" channel and Δ_2 is a "stuck-at-1" channel, as shown in Figure 2. Furthermore, assume that

$$P\{S_i^{(r)}=0\}=1-\theta,\ P\{S_i^{(r)}=1\}=\theta_1,\ P\{S_i^{(r)}=2\}=\theta_2,\quad i=1,\ldots,L$$

with $\theta=\theta_1+\theta_2$, and

$$P\{S_j^{(c)}=0\}=1-\phi,\ P\{S_j^{(c)}=1\}=\phi_1,\ P\{S_j^{(c)}=2\}=\phi_2,\quad j=1,\ldots,M$$

with $\phi=\phi_1+\phi_2$. The cell of the i-th row and the j-th column is modeled as one of the component channel whose state $S_{i,j}$ is determined by $S_i^{(r)}$ and $S_j^{(c)}$ in the manner described in Section II and shown in Table 1. For $S_{i,j}=0$, the equiprobable input achieves cell capacity of $C_o=1-H(p_o)$; and for $S_{i,j}=1$ and $S_{i,j}=2$, every input distribution achieves the same cell capacity of $C_1=0$. Therefore, the compatibility assumption holds for this channel model with equiprobable input. Thus (3.9), (3.12), and (3.13) hold. independent identically distributed input achieves capacity of the channel with row and column side information. Arimoto/Blahut's algorithm is used in computing capacities of the channels without side information with row side information only, and with column side information only. Computed results are shown in Table 2. Note that $C_r(L, M)$ is independent of L and $C_c(L, M)$ is independent of M. For large L and M, computational requirement of $C(L, M)$ becomes large. However, these capacities $C(L, M)$, $C_r(L, M)$ and $C_c(L, M)$ are less than $\overline{C}$ but are very close to $\overline{C}$ for large L and M.

Example 2. (A RAM with row and column defects)

We can further simplify the channel model of RAM in the Example 1 to a two-state channel model. The difference between this two-state channel model and the three-state channel model lies in number of channel states and symmetry of component channels. Let us assume that a RAM consists of L rows and M columns of cells; that $\Omega=\{0,1\}$; and that Δ_0 is a binary symmetric channel (BSC) with crossover probability p_o, and Δ_1 is also a BSC with crossover probability $\frac{1}{2}$ as shown in Figure 3. Furthermore, assume that

$$P\{S_i^{(r)}=0\}=1-\theta,\ P\{S_i^{(r)}=1\}=\theta,\quad i=1,\ldots,L,\ \text{and}$$

$$P\{S_j^{(c)}=0\}=1-\phi,\ P\{S_j^{(c)}=1\}=\phi,\quad j=1,\ldots,M.$$

The cell of the i-th row and the j-th column is modeled as one of the component channel whose state $S_{i,j}$ is determined by $S_i^{(r)}$ and $S_j^{(c)}$ in the manner described in Section II and shown in Table 3. For $S_{i,j}=0$, the equiprobable input achieves cell capacity of $1-H(p_o)$; and for $S_{i,j}=1$, every input distribution achieves same cell capacity of 0. Therefore, the compatibility assumption holds for this channel model with equiprobable input. Thus (3.9), (3.12), and (3.13) hold. Since all component channels are BSC, equiprobable input achieves channel capacity for all types of the two-dimensional BI channels. Computed results are shown in Table 4. Note that $C_r(L, M)$ is independent of L and $C_c(L, M)$ is independent of M. For large L and M, computational requirement of $C(L, M)$ becomes large. However these capacities $C(L, M)$, $C_r(L, M)$ and $C_c(L, M)$ are less than $\overline{C}$ but are very close to $\overline{C}$ for large L and M.

Example 3. (A general two-dimensional BI channel)

Let us assume that a two-dimensional BI channel consists of L rows and M columns of cells; that $\Omega=\{0,1,\cdots,J-1\}$; that Δ_0, Δ_1, $\cdots$ and Δ_{J-1} are BSCs with crossover probabilities p_0, p_1,

$\cdots$ and p_{J-1} respectively; and that $p_0 < p_1 < \cdots < p_{J-1}$. Furthermore, assume that

$$P\{S_l^{(r)} = k\} = \theta_l, \qquad l = 1, \ldots, L \quad \text{with} \quad \sum_{k=0}^{J-1} \theta_k = 1, \text{ and}$$

$$P\{S_m^{(r)} = k\} = \phi_m, \qquad m = 1, \ldots, M \quad \text{with} \quad \sum_{k=0}^{J-1} \phi_k = 1.$$

Let us denote the capacities of Δ_0, Δ_1, $\cdots$ and Δ_{J-1} by C_0, C_1, $\cdots$ and C_{J-1} respectively, then $C_0 > C_1 > \cdots > C_{J-1}$. Then, from (3.9), we have that

$$C_{r,c}(L, M) = \mathrm{E}[C_s]$$

$$= C_{J-1} + \sum_{i=0}^{J-2} (C_i - C_{i+1}) \left\{ \sum_{j=0}^{i} \theta_j \right\} \left\{ \sum_{k=0}^{i} \phi_k \right\} \tag{4.3.1}$$

which is independent of L and M, as expected.

CONCLUSIONS

In this paper we have shown that the capacity of the channel with row side information is less than but approaches the capacity of channel with row and column side information, as the the number of rows and columns increases. Similarly for the capacity of the channel with column side information. We have also derived a simple formula for calculating the capacity of a random access memory when side information is available and errors affect whole rows and/or columns.

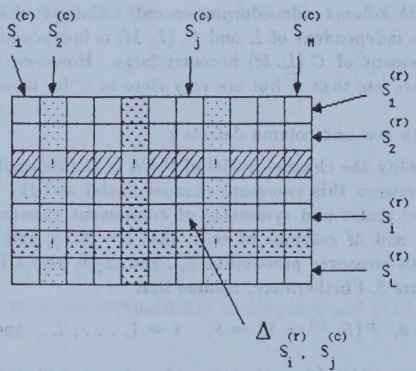


Figure 1. Channel with two-dimensional BI

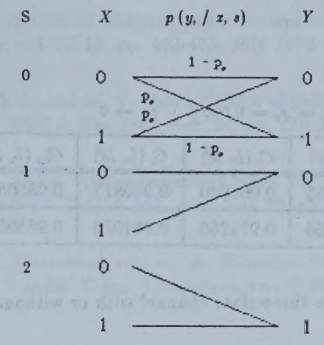


Figure 2. Three-state channel model for a binary memory cell with defects

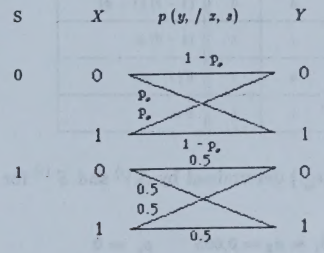


Figure 3. Two-state channel model for a binary memory cell with defects

$S_i^{(r)}$	$S_j^{(c)}$	$S_{i,j}$	$p(s_i^{(r)}, s_j^{(c)}, s_{i,j})$
0	0	0	$(1-\theta)(1-\phi)$
0	1	1	$(1-\theta)\phi_1$
0	2	2	$(1-\theta)\phi_2$
1	0	1	$\theta_1(1-\phi)$
1	1	1	$\theta_1\phi_1$
1	2	1	$\theta_1\phi_2 / 2$
		2	$\theta_1\phi_2 / 2$
2	0	2	$\theta_2(1-\phi)$
2	1	1	$\theta_2\phi_1 / 2$
		2	$\theta_2\phi_1 / 2$
2	2	2	$\theta_2\phi_2$

Table 1. $S_{i,j}$ and $p(s_i^{(r)}, s_j^{(c)}, s_{i,j})$ determined by $S_i^{(r)}$ and $S_j^{(c)}$ for the three-state channel model

$\theta_1 = \theta_2 = 0.01 \quad \phi_1 = \phi_2 = 0.005 \quad p_o = 0$

L	M	$C(L, M)$	$C_r(L, M)$	$C_c(L, M)$	$C_{r,c}(L, M)$
1	1	0.936455	0.965040	0.949812	0.985050
2	2	0.948866	0.971260	0.961076	0.985050

Table 2. Capacities of the three-state channel with or without side information

$S_i^{(r)}$	$S_j^{(c)}$	$S_{i,j}$	$p(s_i^{(r)}, s_j^{(c)}, s_{i,j})$
0	0	0	$(1-\theta)(1-\phi)$
0	1	1	$(1-\theta)\phi$
1	0	1	$\theta(1-\phi)$
1	1	1	$\theta\phi$

Table 3. $S_{i,j}$ and $p(s^{(r)}, s^{(c)}, s_{i,j})$ determined by $S^{(r)}$ and $S^{(c)}$ for the two-state channel model

$\theta_1 = \theta_2 = 0.01 \quad \phi_1 = \phi_2 = 0.005 \quad p_o = 0$

L	M	$C(L, M)$	$C_r(L, M)$	$C_c(L, M)$	$C_{r,c}(L, M)$
1	1	0.936455	0.965040	0.949812	0.985050
2	2	0.945089	0.969382	0.957347	0.985050
3	3	0.951962	0.972491	0.962789	0.985050
4	4		0.974753	0.966769	0.985050
5	5		0.976426	0.969725	0.985050
6	6		0.977687	0.971960	0.985050

Table 4. Capacities of the two-state channel with or without side information

ACKNOWLEDGEMENT

This research was supported by the National Science Foundation under grant ECS-8307150.

REFERENCE

[1] R. J. McEliece and W. E. Stark, "Channels with block interference," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 44-53, Jan. 1984.

[2] R. J. McEliece, *The Theory of Information and Coding*. Addison-Wesley, 1977.

[3] S. Arimoto, "An algorithm for computing the capacity of arbitarary discrete memoryless channels," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 14-20, Jan. 1972.

[4] R. E. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 460-473, July 1972.

[5] D. C. Bossen, C. L. Chen, and M. Y. Hsiao, "Fault alignment exclusion for memory using address permutation," *IBM J. Res. Dev.*, vol. 28, pp. 170-176, March 1984.

[6] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-art review," *IBM J. Res. Dev.*, vol. 28, pp. 124-134, March 1984.

[7] W. F. Mikhail, R. W. Bartoldus, and R. A. Rutledge, "The reliability of memory with single-error correction," *IEEE Trans. Computers*, vol. C-31, pp. 560-564, June 1982.

THE CAPACITY OF JAMMED COMMUNICATIONS CHANNELS USING SPREAD SPECTRUM MODULATION

BRIAN HUGHES and PRAKASH NARAYAN
Electrical Engineering Department
University of Maryland
College Park, Maryland 20742

I. INTRODUCTION

Spread spectrum systems are a collection of modulation techniques which are effective in suppressing intentional interference, e.g., jamming, as also unintentional interference. These techniques usually fall into one of the following three classifications: direct sequence (DS) methods, time hopping (TH) methods, and frequency hopping (FH) methods. Hybrid techniques formed by combining the methods mentioned above are also used. A detailed discussion of spreading techniques can be found in Dixon [1] and Pickholtz, et al [2].

In the past, attempts to quantify the interference suppression of spread spectrum systems have been system specific and usually have assumed that the jammer is artifically confined to some canonical jamming technique such as "partial band" or "tone" jamming (see, however, the thesis by Stark [3]). At present, a general framework for the study of jammed, spread spectrum systems is still lacking (see, however, Ericson [4]).

In this paper, such a framework is proposed. In section II, a general, abstract model of jammed, spread spectrum communications is introduced. The model is general in the sense that it accommodates all common spreading techniques (e.g. FH, TH, DS and hybrid techniques), any carrier modulation scheme, and "hard" or "soft" decision demodulation. In section III, the capacity of this system is determined. This result provides a systematic method by which the most effective jamming threat to any spread spectrum system can (at least in principle) be determined.

II. MODELS FOR JAMMED, SPREAD SPECTRUM SYSTEMS

Consider first the problem of modeling the effect of jamming on an "ordinary" channel (no spread spectrum). Assume, for simplicity, discrete channel inputs, $x \in X$, and outputs, $y \in Y$ (the generalization to infinite alphabets will be obvious). Without jamming, a memoryless channel is conventionally modeled by a set of fixed transition probabilities, $w(y|x)$. These probabilities can (in principle) be measured and used by the transmitter to construct codes tailored to the channel.

The situation is quite different when a jammer is present. Let s, taking values in Σ, index all the signals, $\eta_s(t)$, that the jammer is capable of producing. When the jamming signal $\eta_s(t)$ (unknown to the transmitter) is applied to the channel, the transition probabilities are altered, say to $w(y|x,s)$. In order to communicate reliably without prior knowledge of $\eta_s(t)$, the transmitter must construct robust codes which perform reliably against every jamming signal. This is equivalent to requiring these codes to perform reliably for all channels in

This research was sponsored by the Office of Naval Research under grant N00014-84-6-0101.

$$\{w(y|x,s) \mid x\varepsilon X,\ s\varepsilon\Sigma,\ y\varepsilon Y\} \tag{2.1}$$

This model of a jammed communications channel, called and arbitrarily varying channel (AVC), was first studied by Blackwell, et al [5] and later by Ahlswede [8] (see also Wolfowitz [6]).

A typical spread spectrum modulation scheme has the following structure. Prior to each channel use the transmitter and receiver select from a "large" pool of potential carrier signals, a "small" subcollection for use in the given transmission. In effect, one "instantaneous" channel (or subchannel) is selected from a collection of many available channels. The selection of carrier signals is independent of the source message (the receiver does not yet know the message) and is ideally performed in a manner which cannot be predicted by the jammer.

The jammed, spread spectrum channel can be modeled in the following way. Let the variable $z\varepsilon Z$, hereafter called the key (after Ericson [4]) index all possible subcollections of carrier signals employed in the given spreading technique. For example, z might index the carrier frequency in FH, the transmission time in TH, or the chip sequence in DS. To each key choice, $z\varepsilon Z$, there corresponds an arbitrarily varying channel, $\{w(y|x,z,s)|s\varepsilon\Sigma\}$, reflecting the jammer's effect on the instantaneous channel formed by the carriers which correspond to z. No loss of generality results from the assumption that the input and output alphabets, respectively X and Y, are the same for each key choice. The jammed, spread spectrum channel can then be modeled by an object which will be called a controlled AVC (CAVC),

$$\{w(y|x,z,s) \mid y\varepsilon Y,\ x\varepsilon X,\ z\varepsilon Z,\ s\varepsilon\Sigma\} \tag{2.2}$$

The CAVC is a channel with three inputs x, z, and s, and one output, y. The first two inputs are selected by the transmitter; the symbol x is used to represent the source, while the key, z, is chosen independently of the source. As before, s indexes the jammer's signals. The interpretation is that the transmitter and receiver agree in advance that z will not be used to convey source information, but are otherwise free to select z in a way that maximizes the information conveyed by x to y.

III. CODING THEOREM

A block code for the jammed, spread spectrum channel (or CAVC) is a pair,

$$\begin{aligned} C_n &= \{C_n,\ \nu_n\} \\ C_n &= \{(u_i,\ A_i) \mid 1\leq i\leq M\} \end{aligned} \tag{3.1}$$

where $\{u_i\}_{i=1}^{M}$ are codewords in X^n, the decoding sets $\{A_i\}_{i=1}^{M}$ partition Y^n, and ν^n is a probability distribution for the key sequence $z^n\varepsilon Z^n$.

The choice of performance measures appropriate for this code will depend on the assumed abilities of the jammer. We will allow the jammer knowledge of the transmitter's codebook and all previously transmitted blocks; however, we will not allow him to use information obtained earlier within the same block. In particular repeat-back and frequency-follower jammers are not allowed.

These assumptions lead to three measures of the performance of C_n: the usual information rate,

$$R_s(C_n) = \frac{\ell n(M)}{n}, \tag{3.2a}$$

the worst case error probability,

$$\lambda(C_n) = \sup_{s^n \in \Sigma^n} \frac{1}{M} \sum_{i=1}^{M} \sum_{z^n \in Z^n} w^n(A_i^c | u_i, z^n, s^n) \nu^n(z^n), \tag{3.2b}$$

and a new quantity (introduced by Ericson [4]) called the key rate,

$$R_k(C_n) = \frac{H(\nu^n)}{n} \tag{3.2c}$$

where,

$$H(\nu^n) \triangleq - \sum_{z^n \in Z^n} \nu^n(z^n) \ell n \left[\nu^n(z^n)\right]$$

The latter measure is justified by Shannon's First Theorem (e.g. Gallagher [7]) from which it follows that $R_k(C_n)$ is the minimum average number of bits per channel use which must be shuttled between (or stored by) the transmitter and receiver to keep each informed of carrier choices. Note that $R_k(C_n) = 0$ corresponds to a periodic key sequence.

A rate pair (R_s, R_k) is achievable if for every $\varepsilon > 0$ there is, for sufficiently large n, a code C_n for which,

$$R_s(C_n) \geq R_s - \varepsilon$$

$$R_k(C_n) \leq R_k$$

$$\lambda(C_n) \leq \varepsilon$$

In order to state the coding theorem for the CAVC, it is necessary to introduce some additional notation. For all $n \geq 1$ and any probability measures ν^n, σ^n, and π^n on Z^n, Σ^n, and X^n respectively, define[1]

$$w^n_{\sigma^n}(y^n | x^n, z^n) \triangleq \sum_{s^n} w^n(y^n | x^n, z^n, s^n) \sigma^n(s^n) \tag{3.3a}$$

$$w^n_{\nu^n}(y^n | x^n, s^n) \triangleq \sum_{z^n} w^n(y^n | x^n, z^n, s^n) \nu^n(z^n) \tag{3.3b}$$

$$w^n_{z^n}(y^n | x^n, s^n) \triangleq w^n(y^n | x^n, z^n, s^n) \tag{3.3c}$$

1 To simplify notation, we have assumed here that Σ^n and Z^n are discrete; if not, replace (3.3a) and (3.3b) with Lesbesgue-Stieljes integrals.

$$w^n_{\pi^n}(y^n|z^n, s^n) \triangleq \sum_{x^n} w^n(y^n|x^n, z^n, s^n)\pi^n(x^n) \tag{3.3d}$$

Further, we will use compound subscripts such as $w_{\nu^n\sigma^n}(y^n|x^n)$ defined analogously. A probability measure, ν^n, is said to be pure if all its mass is concentrated at a single point (i.e. $\nu^n(z^n) = 1$ for some z^n). Also define,

$$\tau^n_{z^n}(\pi^n) \triangleq \text{convex closure } \{w^n_{\pi^n z^n}(\cdot|s^n)\,|\,s^n \varepsilon \Sigma^n\}$$

$$\tau^n_{\nu^n}(\pi^n) \triangleq \text{convex closure } \{w^n_{\pi^n \nu^n}(\cdot|s^n)\,|\,s^n \varepsilon \Sigma^n\}$$

Define the mutual information for any n-ary channel $w^n(y^n|x^n)$ and any $\pi^n(x^n)$ to be,

$$I(\pi^n, w^n) \triangleq H(w^n_{\pi^n}) - \sum_{x^n} H(w^n_{x^n})\pi^n(x^n) \tag{3.4}$$

where $H(\cdot)$ is defined as in (3.2c).

Definition: The CAVC (2.2) has the random separation property (RSP) if for some $n \geq 1$, and some probability measure ν^n on Z^n, there are two distributions π^n and $\tilde{\pi}^n$ on X^n for which,

$$\tau^n_{\nu^n}(\pi^n) \bigcap \tau^n_{\nu^n}(\tilde{\pi}^n) = \phi \tag{3.5}$$

The CAVC has the deterministic separation property if ν^n in (3.3) can be chosen to be pure.

The following theorem is the main result of this paper.

Theorem 1 (Coding Theorem for the CAVC). For the CAVC (2.2) define,

$$C_d \triangleq \begin{cases} \max_{\pi,z} \inf_{\sigma} I(\pi, w_{z\sigma}) & \text{, if DSP holds} \\ 0 \text{ ,} & \text{otherwise} \end{cases} \tag{3.6a}$$

$$C_r \triangleq \begin{cases} \max_{\pi,\nu} \inf_{\sigma} \sum_{z\varepsilon Z} I(\pi, w_{z\sigma})\nu(z), & \text{if RSP holds} \\ 0 \text{ ,} & \text{otherwise} \end{cases}$$

Then the set of all achievable rate pairs, R^*, is given by

$$R^* = R_d \cup R_r$$

where,

$$\mathcal{R}_d \triangleq \{(R_s, R_k) \mid 0 \le R_s \le C_d,\ R_k = 0\}$$
$$\mathcal{R}_r \triangleq \{(R_s, R_k) \mid 0 \le R_s \le C_r,\ R_k > 0\} \qquad (3.6b)$$

To illustrate the model and the coding theorem, two examples are considered: FH/BPSK and DS/BPSK, both against a jammer of bounded energy. The results rigorously confirm certain spread spectrum "folk theorems" while disproving other widely held beliefs.

IV. ACKNOWLEDGEMENT

The authors would like to thank Anthony Epremides for many valuable discussions on this problem. The first author has also benefited from discussions with Jeffrey Wieselthier and Dennis MacGregor of the Naval Research Laboratory.

V. REFERENCES

[1] R.C. Dixon; Spread Spectrum Systems, John Wiley and Sons (New York, 1976).

[2] R.L. Pickholtz, D.L. Schilling, and L.B. Milstein "Theory of Spread Spectrum Communications - A Tutorial," IEEE Trans. Commun. COM-30, 855-884, May 1982.

[3] W.E. Stark; "Coding for frequency-hopped spread-spectrum channels with partial band interference," Coordinated Science Laboratory Report R-945 (also Ph.D. Thesis), University of Illinois, July 1982.

[4] T. Ericson; "The Non-cooperative Binary Adder Channel," preprint of paper presented at the International Symp. on IT, St. Jovite, Quebec, Canada, Sept. 1983.

[5] D. Blackwell, L. Breiman, and A.J. Thomasian, "The Capacities of Certain Channel Classes Under Random Coding," Ann. Math. Stat., Vol. 31, 558-567, (1960).

[6] J. Wolfowitz; Coding Theorems of Information Theory, 3rd Ed., Springer Verlag (Berlin, 1978).

[7] R.G. Gallagher; Information Theory and Reliable Communication, John Wiley and Sons (New York, 1968).

[8] R. Ahlswede; "Elimination of Correlation in Random Codes for Arbitrarily Varying Channels," Z. Wahrscheinlichkeitstheorie verw. Gebiete, Vol. 44, pp. 159-175 (1978).

A RESULT ON THE CHARACTERIZATION OF THE THREE-DIMENSIONAL CAPACITY REGIONS, WITH AND WITHOUT FEEDBACK, OF THE BINARY ERASURE MULTIPLE-ACCESS CHANNEL

Kristien DE BRUYN and Edward C. VAN DER MEULEN
K.U.Leuven, Dept. of Mathematics, 3030 Heverlee, Belgium.

For the discrete memoryless (d.m.) binary erasure multiple-access channel (BEMAC), van der Meulen [1] showed that the symmetrical rate pair (.79113,.79113) is achievable in the classical feedback situation. Willems [2] proved that this rate point is on the boundary of the corresponding two-dimensional feedback capacity region. In this contribution we extend this result to the three-dimensional Slepian-Wolf communication situation, both with and without feedback, by determining the boundary lines $R_1 = R_2$, of both capacity regions.

The d.m. BEMAC has two binary input alphabets ($\mathcal{X} = \mathcal{Y} = \{0,1\}$), a ternary output alphabet ($\mathcal{Z} = \{0,1,2\}$) and its transition matrix is defined by

$$w(z|x,y) = i \quad \text{iff} \quad x + y = z.$$

We consider the communication situation introduced by Slepian and Wolf [3], whereby messages from three independent sources M_o, M_1, M_2 are encoded by two encoders, as pictured in Fig. 1.

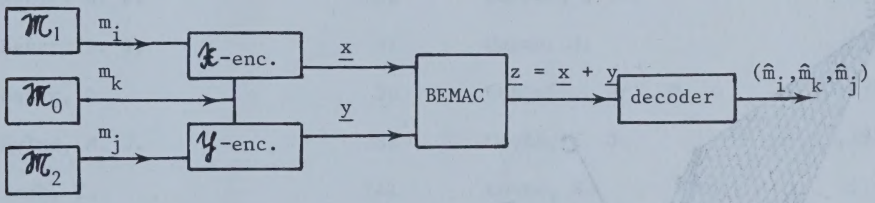


Fig. 1. The BEMAC in the Slepian-Wolf configuration.

The capacity region is then given by

$$C = \bigcup_{|\mathcal{U}| \leq 6} \bigcup_{P_{SW}} \{(R_o,R_1,R_2) \in \mathbb{R}_+^3 \mid R_1 \leq H(X|U),\ R_2 \leq H(Y|U),\ R_1+R_2 \leq H(Z|U),\ R_o+R_1+R_2 \leq H(Z)\},$$

whereby P_{SW} is the set of probability distributions on $\mathcal{U} \times \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ such that

$$p(u,x,y,z) = p(u)p(x|u)p(y|u)w(z|x,y).$$

If moreover feedback is available to one or both encoders, then [4]

$$C^F = \bigcup_{|\mathcal{U}| \leq 5} \bigcup_{P_{SW}} \{(R_o,R_1,R_2) \in \mathbb{R}_+^3 \mid R_1 \leq H(X|U),\ R_2 \leq H(Y|U),\ R_o+R_1+R_2 \leq H(Z)\}.$$

In this contribution the points $(R_o,R,R) \in C$ (respectively C^F) are characterized. This problem is of interest since in general the computation of boundaries of capacity regions is considered to be hard.

This work was partially supported by Research Project GOA 83/88-53 of the Belgian Ministerie van Wetenschapsbeleid, Brussels.

In the results, which now follow, we refer to Fig. 2. below. The (R_0,R,R) boundary curve of C^F is EAB, consisting of the line segment EA on the total cooperation plane, i.e., on which $R_0+2R = \log_2 3$, and the curve segment

$$AB = \{(R_o,R,R) = (f(\alpha)-2h(\alpha),h(\alpha)) \mid \alpha \in [.2113,.2377]\},$$

while the (R_o,R,R) boundary curve of C is ECD, given by the line segment EC on the total cooperation plane and the curve segment

$$CD = \{(R_o,R,R) = (f(\alpha)-2h(\alpha)-2\alpha(1-\alpha),h(\alpha)-\alpha(1-\alpha),h(\alpha)-\alpha(1-\alpha)) \mid \alpha \in [.2113,.2377]\}.$$

Hereby $h(\alpha) = -\alpha\log\alpha-(1-\alpha)\log(1-\alpha)$ and $f(\alpha) = h(2\alpha(1-\alpha))+\alpha^2+(1-\alpha)^2$. We found A = (.0970,.7440,.7440) and C = (.4304,.5773,.5773). As follows from [1] and [2], B = (0,.79113,.79113) is on the boundary of C^F and D = (0,.75,.75) is well-known to lie on the boundary of C. The point E = $(\log_2 3,0,0)$ belongs to the boundary of both capacity regions.

The methods used to derive those results are similar to the ones in [2] and involve convexity arguments and optimization techniques.

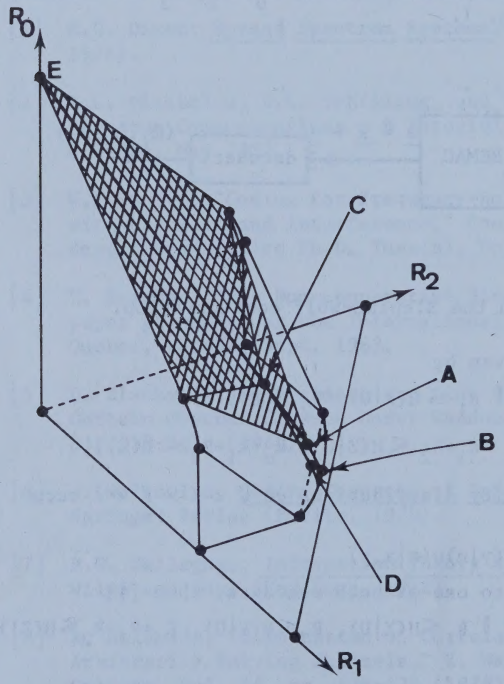


Fig. 2. Sketch of the three-dimensional capacity regions C and C^F, with exactly determined boundary lines $R_1 = R_2$.

REFERENCES

[1] E.C. van der Meulen, "A survey of multi-way channels in information theory : 1961-1976", IEEE Trans. Inform. Theory, vol IT-23, no. 1, pp. 1-37, Jan. 1977.

[2] F.M.J. Willems, "Two results for the multiple-access channel with feedback", Proc. of the Fourth Symposium on Information Theory in the Benelux, Haasrode, Belgium, E.C. van der Meulen ed., Acco Leuven, pp. 189-198, 1983.

INDEX OF AUTHORS

Abaya, E. F. 579
Abutaleb, A. 142
Afek, Y. 689
Aggarwal, A. 81
Akl, S. G. 946
Akman, V. 103
Allebach, J. P. 798
Allen, G. H. 878
Arce, G. R. 162,172
Atallah, M. J. 101,113
Atkin, G. E. 711
Baker, B. S. 91
Baker, D. 50
Balas, M. J. 232
Beck, T. 741
Bender, D. J. 642
Bentley, J. L. 233
Berman, F. 921
Beuerman, S. L. 395
Beutler, F. J. 603
Bhattacharya, B. 120
Bhattacharyya, S. P. 755
Biglieri, E. 901
Bisdikian, C. 385
Bitmead, R. R. 878
Blake, I. F. 711
Boorstyn, R. 20
Bozzo, C. 441,956
Brzakovic, D. 792
Bueno, G. 441
Caceres, A. 553
Carroll, R. L. 652
Chang, C.-K. 931
Chaparro, L. F. 808
Chazelle, B. 71
Chen, D. S. 798
Chen, M.-S. 20
Cheung, J. Y. 132
Chirlian, P. M. 421
Cohoon, J. P. 488
Conan, J. 559
Costello, Jr., D. J. 535
Coyle, E. J. 182,395
Cunto, W. 510
Dandapani, R. 827
DeBruyn, K. 975,997
DeCarlo, R. A. 192,757
Delp, E. J. 782
Deng, H. 535
De Oliveira Duarte, A. M. 741
DePano, N. A. A. 81
Dickinson, B. W. 299
Dunne, P. E. 911
Eichel, P. H. 782
Elliott, D. L. 753
Emre, E. 364,376
Ephremides, A. 11,50
Erickson, K. T. 431

Fernandez-Gaucherand, E. 132
Ferreira, C. 597
Ficola, A. 225
Fincham, W. F. 876
Fischer, T. R. 569
Fitch, J. P. 182
Flentz, J. M. 889
Fortune, S. J. 91
Fouques, M. 956
Fowell, R. A. 642
Franklin, W. R. 103
Frederickson, G. N. 271
Frieze, A. 663
Fu, S. J. 221
Fuhrmann, D. R. 891
Gafni, E. 689,699
Gajic, Z. 223
Gallagher, N. C. 182
Georghiades, C. N. 739
Gertler, J. 597
Ghosh, B. K. 625
Gonzalez, T. F. 498
Gouda, M. G. 931
Guerra, C. 965
Haddad, M. A. 940
Halverson, D. R. 328
Hanna, M. T. 469
Heegard, C. 730
Hemachandra, L. A. 545
Hero, A. 344
Herro, M. A. 743
Hirschberg, D. S. 478,508
Howze, J. W. 755
Hsieh, J. G. 232
Hu, L. 743
Huang, N. K. 775
Huang, Y. F. 889
Hughes, B. L. 992
Jamieson, L. H. 765
Johnson, M. 560
Kailath, T. 1
Kalson, S. 333
Kanefsky, M. 808
Karlsson, R. G. 264
Kelly, P. A. 318
Kennedy, Jr., P. J. 374
Kershenbaum, A. 20
Khorrami, F. 211
Kleinrock, L. 60
Knowles, G. 376
Koch, D. B. 459
Korfhage, W. 699
Kumar, P. R. 587
Larmore, L. L. 478,508
Lea, C. T. A. 413
Lee, C. C. 354
Lee, E. B. 615
Lee, J. H. 983
Lee, S.-L. 498
Lee, T.-A. 730

Li, S. 299
Liakopoulos, A. 792
Liestman, A. L. 701
Lindner, D. K. 745
Lindstrom, E. E. 525
Liu, B. 450,891
Lloyd, E. L. 291
Loeloeian, M. 559
Longley, L. A. 354
Lu, W.-S. 615
Lucertini, M. 405
Mahaney, S. R. 91
Majdi, S. O. 755
Marino, R. 225
Martin, K. 309
Mathews, V. J. 152
Matsumoto, T. 201
Matthews, G. P. 192
Maxwell, P. C. 837
McLoughlin, M. P. 162
Meditch, J. S. 413
Merakos, L. 385
Michel, A. N. 431
Mintz, M. 309,588
Moreno, O. 553
Munro, J. I. 510
Murgolo, F. D. 516
Narayan, P. 992
Nerella, R. 720
Nicosia, S. 225
Ohta, Y. 201
Olson, K. W. 817
Onural, L. 794
Oommen, B. J. 243
O'Reilly, J. J. 741
O'Sullivan, J. A. 864
Otoo, E. J. 253
Ozguner, F. 817
Ozguner, U. 211
Paletta, G. 405
Papoutsis, T. 808
Pearlman, W. A. 560
Peters, J. 671
Prusinkiewicz, P. 796
Puthenpura, S. 613
Radhakrishnan, D. 844
Ramamoorthy, P. A. 940
Ramarao, K. V. S. 681
Rao, S. K. 1
Rey, C. 122
Richards, D. 701
Ribble, E. 817
Root, W. L. 318
Rosier, L. E. 931
Ross, K. W. 603
Rozsa, P. 763
Rudolph, L. 663,671
Ryan-Schuck, E. 588
Sain, M. K. 864
Saluja, K. K. 827

Sawan, M. E. 221
Schmeck, H. 946
Schwartz, S. 344
Scott, P. D. 794
Sebok, D. R. 757
Selbuz, H. 761
Shafai, B. 652
Shannon, G. 921
Sharma, R. 844
Sibul, L. H. 887
Siffredi, E. 956
Siljak, D. D. 201
Silvester, J. A. 30
Simaan, M. 469
Singh, A. D. 854
Sinha, N. K. 613,763
Sleator, D. D. 233
Soffa, M. L. 291
Smith, K. D. 765
Snyder, D. L. 343,739
Srinivas, M. A. 271
Stark, W. E. 983
Storey, J. S. 40
Sullivan, B. J. 450
Taghizadeh, K. 720
Tague, J. A. 887
Tarjan, R. E. 233
Tobagi, F. A. 40
Toy, M. 421
Tran, M. T. 221
Tranter, W. H. 459
Tseng, H.-C. 569
Turner, J. S. 281
van der Meulen, E. C. 997
Viswanathan, R. 720
Vitter, J. S. 525
Wang, C.-C. 291
Wang, J. K. 817
Wang, S.-H. 874
Ward, R. K. 122
Warter, P. J. 172
Wei, V. K. 233,545
Wieselthier, J. 50
Wise, G. L. 328,579
Yan, T. Y. 901
Yao, K. 333,901
Yaz, E. 599,761
Youn, D. H. 152
Yu, K. K. 152
Yurtseven, M. K. 876
Zak, S. H. 192,615
Zhang, S.-Y. 635,759
Zhu, R. 11

UNIVERSITY OF ILLINOIS-URBANA
621.3811AL5 C001
PROCEEDINGS / ALLERTON CONFERENCE ON COM
1984
3 0112 008149194